Question Answering with Hybrid Data and Models

Sanjay Kamath Ramachandra Rao

▶ To cite this version:

Sanjay Kamath Ramachandra Rao. Question Answering with Hybrid Data and Models. Document and Text Processing. Université Paris-Saclay, 2020. English. NNT : 2020UPASS024 . tel-02890467

HAL Id: tel-02890467
https://theses.hal.science/tel-02890467

Submitted on 6 Jul 2020

université
PARIS-SACLAY

Question Answering with Hybrid Data and Models

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580
Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique
Unité de recherche : Université Paris-Saclay, CNRS, LIMSI, 91400, Orsay, France
Référent : Faculté des sciences

Thèse présentée et soutenue à Orsay, le 06/02/2020, par

SANJAY KAMATH RAMACHANDRA RAO

Composition du Jury

Nicolas Sabouret Professeur, Université Paris-Saclay	Président
Patrice Bellot Professeur, Université Aix-Marseille	Rapporteur & Examinateur
Mohand Boughanem Professeur, Université Paul Sabatier	Rapporteur & Examinateur
Catherine Berrut Professeure, Université Grenoble Alpes	Examinatrice
Patrick Gallinari Professeur, Sorbonne Université	Examinateur
Anne Vilnat Professeure, Université Paris-Saclay	Examinatrice
Brigitte Grau Professeure, ENSIIE	Directrice de thèse
Yue Ma Maître de Conférences, Université Paris-Saclay	Co-encadrante de thèse

Thèse de doctorat

NNT : 2020UPASS024

Synthèse en français

La recherche de réponses à des questions relève de deux disciplines : le traitement du langage naturel et la recherche d'information. L'émergence de l'apprentissage profond dans plusieurs domaines de recherche tels que la vision par ordinateur, le traitement du langage naturel, la reconnaissance vocale, etc. a conduit à l'émergence de modèles de bout en bout et les travaux actuels de l'état de l'art en question-réponse (QR) visent à mettre en oeuvre de tels modèles.

Dans le cadre du projet GoASQ[1], l'objectif est d'étudier, comparer et combiner différentes approches pour répondre à des questions formulées en langage naturel sur des données textuelles, en domaine ouvert et en domaine biomédical. Le travail de thèse se concentre principalement sur 1) la construction de modèles permettant de traiter des ensembles de données à petite et à grande échelle, et 2) l'exploitation de connaissances sémantiques pour répondre aux questions par leur intégration dans les différents modèles. Nous visons à fusionner des connaissances issues de textes libres, d'ontologies, de représentations d'entités, etc.

Afin de faciliter l'utilisation des modèles neuronaux sur des données de domaine de spécialité, généralement de petite taille, nous nous plaçons dans le cadre de l'adaptation de domaine. Nous avons proposés deux modèles de tâches de QR différents, évalués sur la tâche BIOASQ de réponse à des questions biomédicales, et nous montrons par nos résultats expérimentaux que le modèle de *Questions-Réponses ouvert*[2] convient mieux qu'une modélisation de type *Compréhension machine*[3], qui est la plus couramment utilisée. Nous pré-entrainons le modèle de *Compréhension machine*, qui sert de base à notre modèle, sur différents ensembles de données pour montrer la variabilité des performances lorsque ces modèles sont adaptés au domaine biomédical. Nous constatons que l'utilisation d'un ensemble de données particulier (ensemble de données SQUAD v2.0) pour la pré-entraînement donne les meilleurs résultats lors du test et qu'une combinaison de quatre jeux de données donne les meilleurs résultats lors de l'adaptation au domaine biomédical. Nous avons effectué des expériences à l'aide de modèles de langage à grande échelle, comme BERT[4], qui sont adaptés à la tâche de réponse aux questions. Les performances varient en fonction du type des données utilisées pour pré-entrainer BERT. Nous en avons conclu que le modèle de langue appris sur des données biomédicales, BIOBERT, constitue le meilleur choix pour le QR biomédical.

[1] https://goasq.lri.fr/

[2] Extraction de la réponse étant donné un ensemble de paragraphes, pertinents et non pertinents

[3] Extraction de la réponse étant donné un paragraphe pertinent

[4] https://github.com/google-research/bert

Étant donné que les modèles d'apprentissage profond visent à fonctionner de bout en bout, les informations sémantiques provenant de sources de connaissances construites par des experts n'y sont généralement pas introduites. Nous avons annoté manuellement et automatiquement un jeu de données par les variantes des réponses de BIOASQ et montré l'importance d'apprendre un modèle de QR avec ces variantes. Nous montrons l'utilité d'exploiter le *Type de réponse attendu et le Types lexical de la réponse* en domaine ouvert et en domaine biomédical par différentes études. Ces types sont ensuite utilisés pour mettre en évidence les entités dans les jeux de données, ce qui montre des améliorations sur l'état de l'art. Par ailleurs l'exploitation de représentations vectorielles d'entités dans les modèles se montre positif pour le domaine ouvert.

Une de nos hypothèses est que les résultats obtenus à partir de modèles d'apprentissage profond peuvent être encore améliorés en utilisant des traits sémantiques et des traits collectifs calculés à partir des différents paragraphes sélectionnés pour répondre à une question. Nous proposons d'utiliser des modèles de classification binaires pour améliorer la prédiction de la réponse parmi les K candidats à l'aide de ces caractéristiques, conduisant à un modèle hybride qui surpasse les résultats de l'état de l'art sur la plupart des ensembles de données.

Enfin, nous avons évalué des modèles de QR ouvert sur des ensembles de données construits pour les tâches de *Compréhension machine* et *Sélection de phrases*. Nous montrons la différence de performance lorsque la tâche à résoudre est une tâche de QR ouverte et soulignons le fossé important qu'il reste à franchir dans la construction de modèles de bout en bout pour la tâche complète de réponse aux questions.

Acknowledgement

I dedicate this thesis to my parents who believed in me and my choice to pursue Computer Science throughout my life since my early school days. Thank you maa, for supporting me everyday and importantly since I moved out of India to study in France. Without your support I would have never finished what I started. Thank you paa, for buying me that amazing PC in my 3rd grade and getting me an internet connection, without that I would never appreciate the world of computers in my childhood.

I sincerely thank my supervisors, Brigitte Grau and Yue Ma who believed in my candidature when I had applied for the role and for supporting me throughout the duration of my PhD, by having regular meetings. You both are great mentors and I will remember these times forever. I would like to thank my jury members Patrice Bellot and Mohand Boughanem for patiently reading my thesis and giving valuable feedback and comments, Patrick Gallinari and Catherine Berrut for accepting to be my examiners, and Nicolas Sabouret for being the president of my jury and hosting the defense. Special thanks to Anne Vilnat for accepting to be my second supervisor because of official reasons after Brigitte's retirement and helping me finish everything easily.

Special thanks to my dear partner Swathisree who stood with me throughout my journey in France, and my friends Karthik B and Amruth AC who are my partners in crime and will always be my partners in crime. Thanks guys.

Last but not the least, my dear colleagues Swen, Zheng, Hicham, Julien, Arnaud, Christopher, Rashedur, Somnath, Sadaf, Anna, Leon Paul, Yuming, Tsanta, Elise, Hanna, Leonardo, Eva, Srikanth, Shreyas, Catherine, Arthur, Charlotte, Pierre, Pavan, and many others more who spent time playing games, being great friends, and guiding me and not making me feel homesick, Thank you all. Permanent members of ILES in LIMSI, specially Patrick, Michael, Aurélie, Gabriel, Cyril, Sahar, Anne-Laure, Sophie and Pierre, thank you all for guiding me and treating me very well during my stay at LIMSI.

Contents

1

Introduction

Question Answering (QA) deals with retrieving relevant answers from documents for a given question. It is a field of research which is an intersection between two major research fields, *Natural Language Processing (NLP)* and *Information Retrieval (IR)*. Unlike document retrieval performed by search engines, question answering relies on extracting suitable short answers which are more specific than lengthy documents which are topic related. Question Answering is often seen as a supervised learning problem which requires labelled data, although some exceptions exist (Lewis et al., 2019).

Question Answering can be defined in several ways based on the target task. Some of Question-Answer pair examples are presented below.

Q: Who is the current President of France?
Document Answer: The President of France, officially the President of the French Republic is the executive head of state of France in the French Fifth Republic. In French terms, the presidency is the supreme magistracy of the country.
The powers, functions and duties of prior presidential offices, as well as their relation with the Prime Minister and Government of France, have over time differed with the various constitutional documents since the French Second Republic. The President of the French Republic is also the ex officio Co-Prince of Andorra, Grand Master of the Legion of Honour and the National Order of Merit. The officeholder is also honorary proto-canon of the Basilica of St. John Lateran in Rome, although some have rejected the title in the past.
The current President of the French Republic is Emmanuel Macron, who succeeded François Hollande on 14 May 2017.

Paragraph Answer: The current President of the French Republic is Emmanuel Macron, who succeeded François Hollande on 14 May 2017.

Short Answer: Emmanuel Macron

A Question-Answer pair with a document, paragraph and short answer.

If the goal of the system is to find a relevant document (with multiple paragraphs) to a query, the task is called as *Document Retrieval*. If the goal is to find a paragraph or a sentence relevant to a query, the task is called as *Answer Selection*. If the goal is to find an answer span in the text, the task is called as *Answer Extraction*. All the examples for each task are shown above for a given question. The target task of the QA system defines the type of output expected from the system.

In the first approaches of QA systems, a question is analysed to determine some elements such as focus (the main entity addressed in the question), question category (factoid, non factoid, summary, yes/no, choice question etc.), and expected answer type (entity type expected from the question). The question terms (words) are used to find documents, paragraphs and sentences which might contain the answer. Answer selection (or extraction or ranking based on how the candidates are chosen) is then performed on the relevant paragraphs or sentences to choose the answer sentence (long) or answer span (short).

The *Information Retrieval (IR)* community focuses on document retrieval and paragraph retrieval, whose role is to fetch a set of relevant supporting texts which might contain the answer for a question. Other models are used for ranking answers candidates based on term frequencies, entities, etc. The *Natural Language Processing (NLP)* community works on the QA topics which use different linguistic features such as named entities, part of speech tags, expected answer types, passage and question representations (different syntactic and semantic approaches) etc. These are used as features for either document or paragraph retrieval models or answer selection/extraction models. Both the communities often use machine learning algorithms to learn different classifiers, ranking functions etc. for their individual fields. (Jouis et al., 2012; Chu-Carroll et al., 2012; Grappy, Grau, et al., 2011)

Domain specific question answering is a special type of question answering that deals with specialized data belonging to a domain. Open domain datasets are usually curated automatically or by using crowd sourced workers, while domain specific datasets are curated with the help of domain experts. Open domain question answering systems must be explicitly adapted to these specific domain data inorder to perform efficiently.

In biomedical domain, there are a plenty of expert curated information available in the form of ontologies, metathesaurus etc. The datasets for biomedical question answering are not large scale in size. Creating a large scale dataset in biomedical domain requires domain expertise which is expensive and a time consuming process. Therefore it is important to adapt an open domain model to biomedical domain.

1.1 Question Answering Pipeline

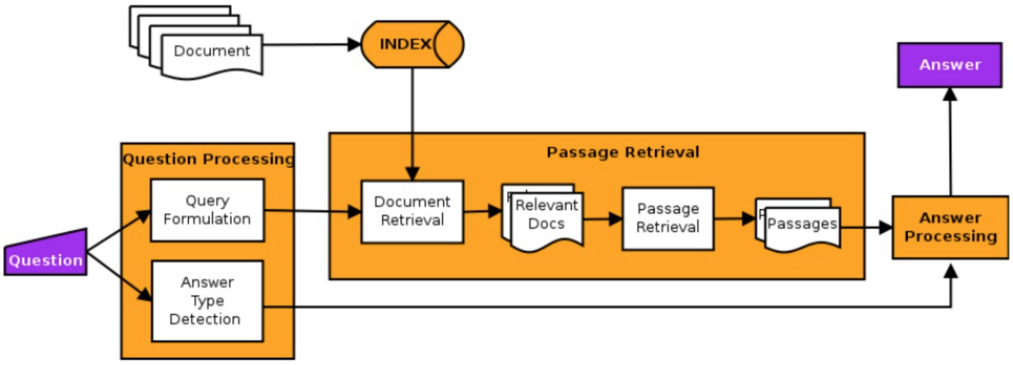


Fig. 1.1: A traditional IR or NLP based question answering pipeline

There are different types of question answering systems based on the type of data they use for processing. Textual data (free text, wikipedia documents, scientific articles, medical text etc.), relational databases, knowledge graphs, ontologies etc. are some of the data sources used for question answering.

A textual question answering system has series of operations either performed sequentially or in parallel. A typical QA pipeline[1] seen in the perspective of Information Retrieval discipline is as shown in the Figure 1.1. The sub-modules of the QA pipeline are explained in brief below.

Question Processing

The question processing module processes a query either to analyse query terms (words) for query expansion, answer type detection etc., or to reformulate it as a SPARQL or a relational database query to find answers in structured databases or ontologies.

The answer type detection analyses the question for different expected answer types based on question words such as "Who", "What", "Where", "When", "How" etc. and determine the expected type of answer such as "Person", "Entity", "Location", "Time",

[1] Figure from: https://web.stanford.edu/class/cs124/lec/watsonqa.pdf

"Procedure" respectively. Based on this type, the answer space can be filtered for better candidate answers.

Passage Retrieval

In a textual QA task, the data used is only textual documents (not databases or knowledge graphs) using which the *Document Retrieval* module applies several techniques to find the relevant documents (more than one paragraphs is termed as a document) for a question, based on the indexed set of documents. These documents are then passed on to a *Paragraph Retrieval* module which finds the best set of paragraphs out of the text documents to extract the answer.

Answer Processing (Long and Short)

The answer processing module processes the paragraphs or sentences in several ways such as ranking candidates based on features (word overlap, entity match, knowledge graph entity types, syntactic analysis, representation matching etc.) and extract answer spans based on named entity type match. This module uses knowledge from databases, ontology, knowledge graphs as reference to extract or find the answers.

The kind of answers such as sentences for long answers or short spans for short answers determines the processing steps involved. For long answers, a *sentence selection* task is used to select an answer sentence. For short answers, an *extraction* task namely *reading comprehension or machine reading or extractive question answering* task is used to extract short answer spans from sentences or paragraphs.

The three answer processing tasks used in our works are:

- *Answer Sentence Selection* - choosing one or more answer sentences or short paragraphs among other candidates, as correct answers for a given question.
- *Reading Comprehension* - extracting short answer spans from a relevant paragraph for a given question.
- *Open Question Answering* - extracting short answer spans from a set of paragraphs (relevant and irrelevant) for a given question.

Since Question Answering is a complex task, each module is considered to be an individual subtask and the focus is on achieving better scores on the subtask target datasets. An issue with pipeline approaches in general, is *Error Propagation*. If an intermediate module causes errors, the error gets propagated onto the final output. The overall error rate would increase. In other words, the error gets propagated across different modules which might reduce the final performance. Evaluating individual module and predicting the overall behaviour in a pipeline approach is cumbersome.

1.2 Redefining the QA Pipeline with Deep Learning

The use of deep learning models in the field of Computer Vision and Speech Processing started to rise rapidly because of the availability of 1) Large scale datasets 2) Access to use hardwares such as GPUs (Graphical Processing Unit) for machine learning. GPUs were earlier used primarily for computer graphics but recently tweaked for training deep learning models, which gave rise to more research interests in the field of machine learning towards deep learning because of the accelerated training times. This was observed in the field of question answering which changed the way some of the traditional QA models work. The intuition of using these models is to 1) avoid handcrafting features for the models, 2) avoid pipeline approaches but rely only on input data to predict the output by automatically learning these features. An ideal deep learning model would be an end-to-end model which relies only on input data and will not have issues like *Error Propagation*.

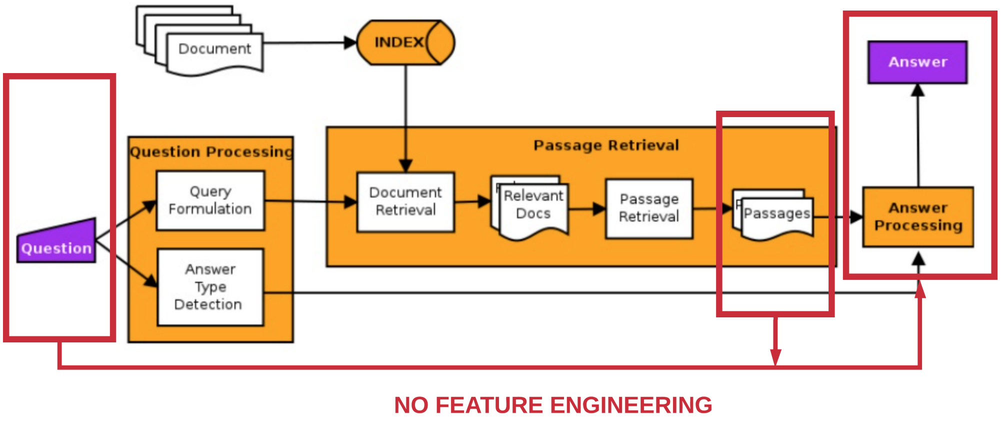


Fig. 1.2: Deep learning QA pipeline (only the modules highlighted in red are used)

The field of *Natural Language Processing (NLP)* witnessed the usage of deep learning models starting with tasks such as Named Entity Recognition (Collobert et al., 2011), Dependency Parsing (Socher et al., 2013), Sentiment Analysis (Moraes et al., 2013) etc. The intuition or the hypothesis behind using deep learning models is that these models are good implicit feature extractors and will not need handcrafted feature engineering which the pipeline modules heavily relied on.

The deep learning models for QA would focus only on the red parts highlighted in the Figure 1.2. An ideal deep learning based QA model would perform answer extraction directly by retrieving the documents, finding the relevant ones, extracting relevant paragraphs and finally extracting the answers out of it. All the modules can be learnt based on the input data and no explicit features would be needed. But building such models has many hurdles to cross.

Hurdles on using deep learning models

Building such a deep learning model (end-to-end model) which requires zero handcrafted features and which learns solely from data has some hurdles.

- Have enough data. (Size)
- Have the right kind of labelled data. (Suitable type)
- Build a single model which does it all. (Architectural Complexity)
- Generalize the model to work on all QA tasks. (Generalization)

Size

For deep learning models to work efficiently, they have to be trained on large scale datasets. Training on small scale datasets will not result in similar performance as on large scale ones. We show this behaviour experimentally in further sections of our work. Because of this behaviour, only tasks which have sufficiently large scale labelled datasets can use deep learning models.

This behaviour causes difficulty in analysing if the model itself is performing poorly or if the dataset is not sufficient for the model to perform better. It causes a dilemma whether or not to design a new model or augment the dataset with more labeled samples and balanced label classes.

To overcome the small size of datasets for biomedical question answering, *Transfer Learning* was first used by (Wiese et al., 2017a). The process involves training models on large scale datasets and fine-tuning the same models on small scale datasets for the same target tasks. When this process is conducted across two or more different domain datasets, it is called as *Domain Adaptation*. The target task remains the same in this approach. Another type of *Transfer Learning* approach is to train models on a task with large scale datasets and modifying final output layer to another target task and fine tuning this model on a small scale dataset of the target task.

Suitable type

The large scale datasets required by the deep neural network models are not just any random datasets. The data should be labelled according to the task for which the model is built. To label such datasets three methods are used in common practice. 1) An automatic annotation method which results in creation of synthetic datasets; 2) Distant supervision methods with some noise; 3) Crowdsourced workers (or sometimes colleagues, students etc.) employed to annotate a dataset which is sufficiently large scale for their model. Although an important highlight is that there is no standard measure to determine a "large scale" dataset.

Architectural complexity

Building an end-to-end model which returns an answer for an input question, is more complex than building submodules, the focus was on using deep learning models on tasks such as *Answer Sentence Selection* and *Answer Extraction* using *Reading Comprehension* by splitting the QA process into two or more sub-tasks. More focus is seen on these topics separately than together into a single big model. The goal of building end-to-end models remained the same but the target tasks became different, that are mainly the sub tasks of the overall QA task.

Even though end-to-end models are the best models one can strive to build for some subtasks, getting certain aspects such as their neural architectures, training approaches, hyperparameters, optimizer functions, random initialization parameters etc. is a big challenge and often sometimes referred as *Architecture Engineering*[2]. From avoiding *Feature Engineering* which the traditional machine learning algorithms

[2] https://smerity.com/articles/2016/architectures_are_the_new_feature_engineering.html

heavily relied on, towards neural networks which heavily relies on *Architecture Engineering*, one problem gives rise to another problem. Techniques like neural architecture search (Elsken et al., 2018) can be used to learn a better architecture than the ones chosen by humans. But they are computationally very expensive to do and bad for environment because the electricity and CO2 emissions they use for this purpose.

Generalization

A common problem working on QA datasets is the usability of these models on other datasets. This problem is not only specific to QA but also other applied machine learning tasks. In other words, *can a model trained on one dataset perform equally well on other datasets on the same or similar task?* This aspect plays a major role in determining if deep learning models trained on large scale datasets can be used to predict on datasets like the domain specific ones which are usually tedious and expensive to curate with the help of experts.

Hurdles in domain specific question answering

Domain specific question answering has several issues in common with some points discussed above. The data is usually curated by domain experts. Contrary to open domain where there is no need of expert knowledge, domain specific data demands domain expertise.

The complexity of the domain expertise makes it hard to obtain labelled datasets of large scale. One such example is the BIOASQ[3] challenge where the biomedical domain experts are asked to annotate certain task datasets for question answering. Over the period of 7 years, the data annotated by human annotators who are domain experts for factoid questions still remain less than 1000 questions.

Domain specific data has two main limitations to be addressed before using the models which work better on open domain data:

Size

Similar to applying deep learning models on small scale datasets, the same problem applies for domain specific datasets which are small scale. Construction of large

[3] http://bioasq.org/

scale dataset is expensive and time consuming because of the domain expertise it demands.

Domain Expertise and Vocabulary

Since the data is domain specific, the vocabulary also consists of new words from a specific domain. For example, using a word embedding space which is trained on Wikipedia might result in missing words from the vocabulary of biomedical domain data. Because the vocabulary is different, detecting named entities, noun phrases, part of speech tags and expected answer types cannot be reused from the traditional QA pipeline used in the open domain setting. There is a requirement for special use case handling for domain specific knowledge integration along with the existing QA models.

While using word embeddings in any NLP tasks, a tokenizer is used to split sentences into words and an embedding vector is found corresponding to the word from a pretrained embedding space. In domains such as biomedical domain or medical domain, the numerical values and co-efficients such as α,β,γ are important and using a tokenizer which tokenizes them differently from the tokenizer used in the pretrained word embeddings, results in missing vocabulary.

Because of the above two limitations, research on domain specific data is significantly less emphasized compared to the open domain. This explains the low results on domain specific tasks like the BIOASQ tasks[4]. Techniques such as domain adaptation and transfer learning can be used to handle this situation, which are relatively easier to do with deep learning models.

Structured Knowledge for Question Answering

Structured knowledge and semantic knowledge by experts provide supporting information for textual data which are enriched, verified, accessible and stored in such a manner that computer applications can access and use it easily with querying languages. They are annotated by human experts or extracted from free text. Relational databases, knowledge graphs and ontologies are some of the examples for such knowledge.

To overcome the missing vocabulary problem in domain specific QA one way is to use domain specific textual resources for training word embedding spaces and

[4] http://participants-area.bioasq.org/results/

another is to use concepts and relationships from ontologies like SNOMED CT[5], UMLS metathesaurus[6] to detect specific entities and their types in the text of QA datasets.

1.3 Research Objectives and Contributions

To address some of the issues discussed in the above sections with respect to deep learning methods, we intend to investigate and compare various question answering system techniques on open domain and domain specific data. Specifically we focus on answer sentence selection task on open domain data, answer extraction by reading comprehension and open question answering tasks on both open domain data and biomedical domain datasets.

Our research questions are the following:

1. How can we build models which work both on small scale and large scale datasets without dropping performance?
2. How can we leverage the structured and semantic knowledge effectively into state of the art question answering models?

In the context of this thesis work - "Question Answering with Hybrid Data and Models", *Hybrid data* refers to using open domain data with specific domain data in a domain adaptation process, plus integrating structured knowledge for annotating training datasets and for enriching the input data. *Hybrid model* refers to an addition of a post processing reranker to account for structured knowledge and collective features obtained from different paragraphs.

Deep learning models have been commonly used in the field of *Natural Language Processing* across various tasks. However, using them on domain specific data which are small scale and are dependent on domain specific vocabulary is not very straightforward. As addressed earlier the performance of these models rely on the *Size* of the datasets.

In Chapter 4, addressing our first research question, we discuss how one can use these models on small scale labelled datasets such as biomedical domain dataset for question answering and discuss different strategies involved in getting better performance. More precisely we discuss details about *Domain adaptation* on biomedical

[5] https://www.snomed.org/snomed-ct/five-step-briefing
[6] https://www.nlm.nih.gov/research/umls/index.html

QA dataset (BIOASQ) by training two task models 1) *Reading Comprehension* and 2) *OpenQA* on open domain QA datasets and fine-tune them on BIOASQ data. We show how a pre-training method using *OpenQA* method suits better for BIOASQ task instead of the other method.

Using deep learning models has become a new trend in the field of research, which has resulted in a lot of outcomes which are incremental in nature. Although often some engineering tricks can improve a model performance slightly (sometimes significantly), research papers tend to not discuss these in detail but rather focus on the mathematical models. Often the performance reported is generalized as the result of the underlying neural network architectures and training methods, but not the subtle difference in preprocessing training data such as using a different tokenizer, different named entity recognition tools, different part of speech taggers, different embedding spaces, highlighting some entity information, using different combination of datasets etc. We present two research articles (Seo et al., 2016; Chen, Fisch, et al., 2017) focused on introducing two different QA models based on RNNs on a same QA task. The authors report the performance on SQUAD dataset leaderboard (Rajpurkar, J. Zhang, et al., 2016). The simple method (Single attention mechanism, faster training) by (Chen, Fisch, et al., 2017) performs better than the complex method (Multiple attention mechanisms, slower training) by (Seo et al., 2016), which shows that the results not always increase based on the increasing complexity.

In Chapter 5, addressing our second research question, we hypothesize that existing QA models can perform better if the input data contain more information than just textual phrases. Structured and semantic knowledge which exist already can be useful for question answering.

Our contributions are:

- *Using entity information* - we explore entity enriched texts for *Reading Comprehension* task on open domain QA dataset.
- *Using answer variants* - we manually annotate more answer spans which are correct besides the gold standard ones in biomedical domain and show how models can perform better when correct data is input.
- *Using expected answer types to highlight entities* - we highlight entities in paragraphs which match the expected answer types from the question using an existing *Answer Sentence Selection* QA model and compare their performance

with baseline model scores and show how using the same model with slightly modified data can improve performance.

- *Improving QA performance using semantic features and structured information for ranking models* - The semantic information from various paragraphs which are provided with the questions in *OpenQA* task are not well modelled in neural network models which consider only a pair of paragraph and question inputs and ignore other paragraph information. In our work we model this collective information and semantic information as features for ranking models to improve QA performance.

The organization of the chapters of this thesis is as shown below: We introduced the context of our research work, presented the introduction to question answering, some hurdles with neural network models in QA and presented our research questions in this chapter.

In the following chapters, we detail the tasks of question answering, datasets used for benchmarking QA models and models which are widely used as state-of-the-art techniques. Chapter 3 presents the overview of QA approaches including the state-of-the-art models widely used these days on different QA tasks. Chapter 4 presents our work on building models for small scale and large scale datasets using domain adaptation. Chapter 5 presents our work on leveraging structured and semantic information into QA models to improve performance.

1.4 Publications

- **2019 - Measuring semantic similarity of clinical trial outcomes using deep pre-trained language representations** - Anna Koroleva, Sanjay Kamath, Patrick Paroubek. Journal of Biomedical Informatics: X, Published in October 2019.

- **2019 - How to Pre-Train Your Model? Comparison of Different Pre-Training Models for Biomedical Question Answering.** - Sanjay Kamath, Brigitte Grau, Yue Ma. Proceedings of the 7th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering. ECMLPKDD, September 2019.

- **2019 - Predicting and Integrating Expected Answer Types into a Simple Recurrent Neural Network Model for Answer Sentence Selection.** - Sanjay Kamath, Brigitte Grau, Yue Ma. 20th International Conference on Computational Linguistics and Intelligent Text Processing - CICLING 2019, April 2019.

- **2018 - An Adaption of BIOASQ Question Answering dataset for Machine Reading systems by Manual Annotations of Answer Spans.** - Sanjay Kamath, Brigitte Grau, Yue Ma. Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering. EMNLP, October 2018.

- **2018 - Verification of the Expected Answer Type for Biomedical Question Answering.** - Sanjay Kamath, Brigitte Grau, Yue Ma. HQA workshop, companion proceedings of the The Web Conference 2018, April 2018.

- **2017 - A Study of Word Embeddings for Biomedical Question Answering.** - Sanjay Kamath, Brigitte Grau, Yue Ma. 4e édition du Symposium sur l'Ingénierie de l'Information Médicale, November 2017.

2 Question Answering - Tasks and Problems

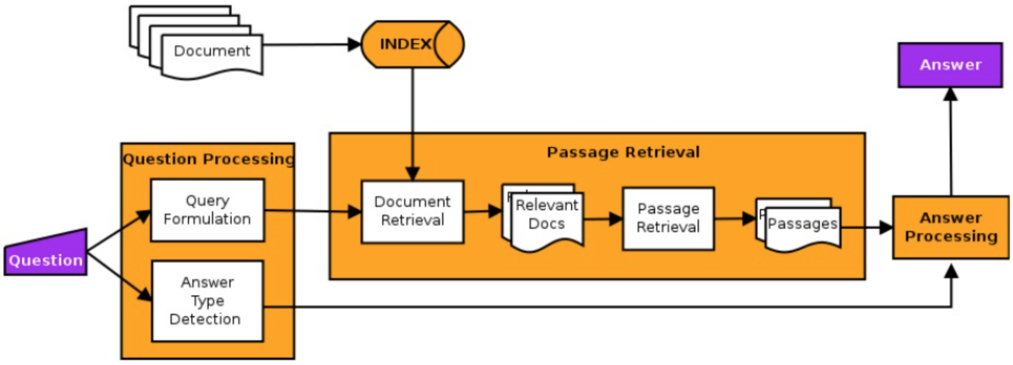


Fig. 2.1: A traditional IR or NLP based question answering pipeline

In this chapter we introduce the task of *Question Answering* formally by explaining different types of tasks, problems, questions, answers and corresponding datasets involved which are being extensively used by the research community.

The question answering pipeline introduced in the Introduction chapter in Figure 1.1 (is also shown in Figure 2.1 for reference) represents the whole pipeline or a complete QA system architecture. Each of these modules individually sometimes are termed as "Question Answering systems" based on different assumptions on the target subtask. Question Answering in different contexts can mean different tasks.

The organization of this chapter is as follows: we first discuss and detail about question answering based on a classification of different types of answers. Then are followed by defining question answering based on different types of tasks, specifically about *Answer Sentence Selection*, *Reading Comprehension* and *OpenQA*. We also highlight and detail some modifications on *Reading Comprehension* task and term it as *RC 2.0*.

We present a list of datasets/corpora used for benchmarking the above mentioned QA tasks and explain some characteristics of open domain and closed domain datasets. Finally we present some evaluation metrics with respect to different tasks and conclude by highlighting what we address in our work.

2.1 Answer based classification

Question Answering systems can be classified based on the answers they expect by analysing the questions. Based on different types of *Answers* or *Results* a system returns to the user, we can briefly classify a QA system into one of the following below:

- Factoid Question Answering
- Non-Factoid Question Answering

Factoid answers are factual answers such named entities, numerical values, lists, currencies, locations etc. For domain specific data such as medical domain the answers can be the names of medicines, diseases, proteins etc.

Q: What country are Volvo automobiles made in? (Location - Country)
A: Sweden

Q: How tall is Mount McKinley? (Numerical value - Height)
A: 6,190 m

Q: What currency is used in Ukraine? (Currency)
A: Ukrainian hryvnia

Q: Who played the role of Heisenberg in the series Breaking Bad? (Person)
A: Bryan Cranston

Factoid QA examples

Non-Factoid answers are answers which are descriptive in nature, such as definitions, procedures, explanation of a phenomenon etc. Often, these answers can be directly extracted from the paragraphs and sometimes a summary can be generated based on textual supporting documents.

Question: Why is ice less dense than water?
Answer Passage: The molecules of water are closer together and constantly moving, whereas the molecules of ice are in a crystal lattice, meaning they're in a rigid formation. When water freezes, the molecules spread out a little more to form the crystal lattice. Since density is mass over volume, and ice has takes up more volume than water, the density of ice is lesser than that of water. Which makes ice float on water.

A non-factoid QA example

Apart from the above two types of QA, there are also Multiple choice QA, Yes/No QA, Conversational QA etc. In our work we only focus on factoid QA therefore the further sections refer to *Factoid Question Answering* as just *Question Answering* or *QA*.

2.2 Task based classification

In the context of factoid question answering as explained above, Question Answering field has seen a lot of different tasks commonly referred to as "Question Answering" although they differ in many ways. These systems can be classified according to their tasks.

2.2.1 Evolution of tasks over time

The tasks have evolved over time based on several factors such as available datasets, approaches, evaluation campaigns etc. The below list of tasks are presented in the chronological order.

1. Open domain question answering based on the pipeline approaches. - Legacy OpenQA, initiated by TREC evaluations.

2. Answer sentence selection - to focus on sentence similarity or textual entailment problems.

3. Reading comprehension and modifications - with neural network approaches.

4. Open domain question answering based on deep learning approaches. - OpenQA

For question answering in an open setting, a question is provided and the system is expected to return an answer. No other supporting information is provided along with the questions. The system has to retrieve a document collection and extract a short answer. This is termed as an *Open QA* task. In the earlier QA systems, pipeline approaches were used for this purpose using information retrieval and NLP techniques, knowledge bases and ontologies. These systems are called as *Legacy OpenQA*. Recent methods use deep learning methods instead of traditional pipeline methods with a goal of building end-to-end models for the same task, these systems are simply called as *OpenQA*.

In between the above two tasks, the focus of QA system was towards *Answer sentence selection* where a sentence among other sentences was to be predicted as a relevant answer or not. Focus further shifted towards answer extraction task from relevant paragraphs which were called as *Reading Comprehension*. There were also systems for multiple choice answers under *Reading Comprehension*.

In this section we define the QA tasks which we address in our work.

2.2.2 Answer Sentence Selection

As per the details presented in Figure 2.1, the task of Answer Sentence Selection deals with finding a sentence as the correct answer for a question. So the *Answer Processing* module determines which paragraph (or sentence) is the correct answer.

Given a question in natural language, the objective of this task is to find relevant sentences among candidate sentences. A sample dataset consists of a Question Q, a set of paragraphs $P = \{p_1, p_2, p_3, ...p_n\}$ and annotated labels either being {0, 1}.

From the definition, the problem can be formulated as a ranking problem, where the goal is to give better rank to the candidate sentences that are relevant to the question. *Pointwise* and *Pairwise* approaches are the two most common approaches to learn the ranking functions.

In the pointwise approach, the ranking problem is transformed into a binary classification problem. More specifically, the training instances are triples (Q_i, P_{ij}, y_{ij}), where Q_i is a question in the dataset, P_{ij} is a candidate answer sentence for Q_i, and y_{ij} is a binary value indicating whether P_{ij} contains the correct answer to Q_i.

In the pairwise approach the ranking function is explicitly trained to score correct candidate sentences higher than incorrect sentences. Given a question, the approach takes a pair of candidate answer sentences and learns to predict which sentence is relevant to the question. Concretely, a score is predicted for each pair of question Q_i and a candidate sentence P_{ij}. We discuss more about several such approaches and the state of the art models in the Section 3.

Q: **Who is the author of the book , "The Iron Lady: A Biography of Margaret Thatcher" ?**
P_1: **The Iron Lady ; a biography of margaret thatcher by hugo young, farrar straus giroux**
P_2: **this second volume of ward's lively and psychologically revealing biography begins with his honeymoon abroad in 1905 and concludes with his election as governor of new york in 1928.**
P_3: **in "The Iron Lady", Young traces the winding staircase of fortune that transformed the younger daughter of a provincial english grocer into the greatest woman political leader since catherine the great.**

An example from TREC-QA dataset, where the sentence P_1 is the correct answer.

The example shown above is from a QA task where the answers are sentences. The models built for this task must determine if an answer sentence is correct or wrong. However, the main limitation of this task is the inability to extract short answers from the sentences. For example, the correct answer for the above example is *Hugo Young*, but the model can only determine if the whole sentence is an answer or contains a short answer in it.

The TrecQA dataset by (M. Wang, Smith, et al., 2007) which was curated from the Trec task[1] is used to benchmark this task.

2.2.3 Reading Comprehension or Machine Reading

To focus on the answer extraction task, the *Reading Comprehension or Machine Reading* task was introduced. By definition, the goal of *Reading Comprehension* task is to "read" a document, which technically means "analyse" or "understand" a document and answer a set of questions based on the document, similar to a *Cloze test*[2] where an exercise consisting of a portion of text with certain items such as entities, words, numbers etc. are removed or hidden from the text. The participant is asked to replace the missing item or items by reading the text. "Cloze tests require the ability to understand context and vocabulary in order to identify the correct

[1] https://trec.nist.gov/data/qa.html
[2] https://en.wikipedia.org/wiki/Cloze_test

language or part of speech that belongs in the deleted parts of the paragraphs." - Wikipedia[2].

Using this intuition, the *Reading Comprehension* task for answer extraction was developed by (Hermann et al., 2015) where a question and a relevant paragraph is the input to a QA system and the answer should be a span (word, set of words, numbers, phrases etc.) from the relevant paragraph. This way an answer from the predicted paragraph can be extracted.

By definition, *Reading Comprehension* task needs comprehension skills to understand and answer a question. The task of QA4MRE[3] (Question Answering for Machine Reading Evaluation) by (Peñas et al., 2013) was held in CLEF QA task, which is termed as *Machine Reading* and had multiple choice answers for questions. The task needed some reasoning and understanding of the paragraph to choose the correct answer among candidate answers and not the extraction of substrings from the paragraphs like in the *Reading Comprehension* task.

In our work we only use and refer to the *Reading Comprehension* task which is formally defined below. A sample dataset consists of a Question Q, a relevant paragraph P along with two string offsets - *Start* and *End* which represent the start and the end character offsets of the answer in the relevant paragraph text. The task of extracting answer span has been generally approached in an uniform manner since its inception by (Hermann et al., 2015). If the answer is a single token, then a probability distribution overall the paragraph terms is used to extract the answer token. If the answer is a span (more than 1 tokens), two classifiers are used to detect the answer span.

A question Q has m terms $Q = \{q_1, ..., q_m\}$ and paragraph S has n terms $S = \{s_1, ..., s_n\}$. An answer span (s, e) is a substring in S. $(s, e) \in S$.

We discuss more about several approaches and the state of the art models in the Chapter 3.

[3] http://nlp.uned.es/clef-qa/repository/qa4mre.php

Question: Which NFL team represented the AFC at Super Bowl 50?

Answer Passage: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24-10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Answer: Denver Broncos. Start Offset: 177

An example of *RC* task from SQUAD dataset (Rajpurkar, J. Zhang, et al., 2016).

The example shown above represents a sample data from a Reading Comprehension dataset SQUAD where usually more than one questions exists per paragraph. A lot of buzz around this task was created by the media,[4] when a system got higher accuracy scores than the human annotator scores. Since then and before this buzz, a lot of works have been published on the SQUAD dataset and their scores are presented on its leaderboard[5] which shows the increasing interests of the community towards this task.

Following this hype, a lot of limitations of the task and the models were put forward by Yoav Goldberg[6] and several others on various social media such as Medium and Twitter.

Following is a list of some of the limitations of Reading Comprehension on SQUAD dataset which gave rise to other methods and datasets later on:

- The answer is guaranteed to be in the paragraph.
- We must find the answer in a given paragraph, not elsewhere.

[4] https://www.wired.com/story/ai-beat-humans-at-reading-maybe-not/, https://www.cnet.com/news/new-results-show-ai-is-as-good-as-reading-comprehension-as-we-are/

[5] https://rajpurkar.github.io/SQuAD-explorer/

[6] `http://u.cs.biu.ac.il/~yogo/on-squad.pdf`

- Annotators see the paragraph when creating the question, resulting in high lexical similarity between question and answer.

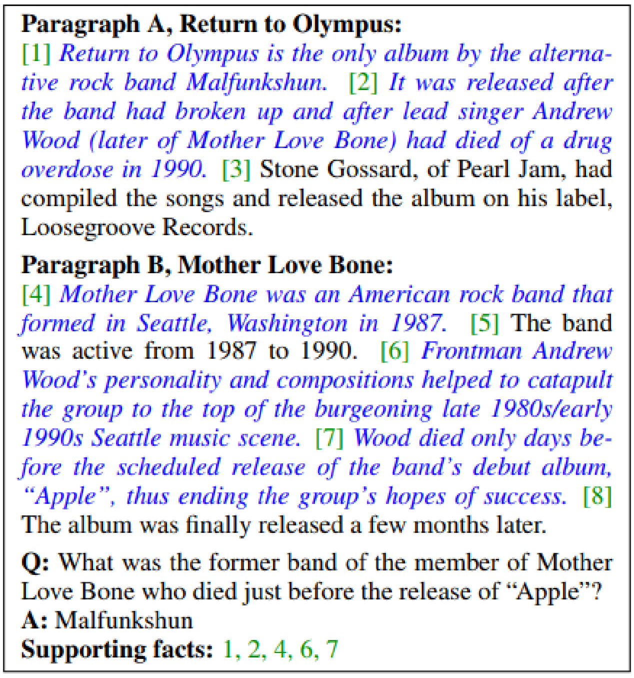
- Adversarial manipulation of the paragraphs (appending an automatically generated noisy sentence to the paragraph) leads to significant drop of performance of many models. (Jia and Liang, 2017)

- No symbolic reasoning is required. Understanding questions to perform reasoning over paragraphs is not possible in traditional datasets such as SQUAD or CNN/Dailymail datasets (Hermann et al., 2015).

2.2.4 Reading Comprehension 2.0 - Modifications

There are several shortcomings from Reading Comprehension task which were listed in the previous section. To address some of them, several works have been published recently. More details about specific works are presented in the Section 3.

In this section we briefly describe three approaches which are often seen as modifications for the *Reading Comprehension* task.



Paragraph A, Return to Olympus:
[1] *Return to Olympus is the only album by the alternative rock band Malfunkshun.* [2] *It was released after the band had broken up and after lead singer Andrew Wood (later of Mother Love Bone) had died of a drug overdose in 1990.* [3] Stone Gossard, of Pearl Jam, had compiled the songs and released the album on his label, Loosegroove Records.

Paragraph B, Mother Love Bone:
[4] *Mother Love Bone was an American rock band that formed in Seattle, Washington in 1987.* [5] The band was active from 1987 to 1990. [6] *Frontman Andrew Wood's personality and compositions helped to catapult the group to the top of the burgeoning late 1980s/early 1990s Seattle music scene.* [7] *Wood died only days before the scheduled release of the band's debut album, "Apple", thus ending the group's hopes of success.* [8] The album was finally released a few months later.

Q: What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?
A: Malfunkshun
Supporting facts: 1, 2, 4, 6, 7

Fig. 2.2: An example from HotpotQA dataset by (Zhilin Yang, Qi, et al., 2018) on multi-hop reasoning for QA.

1. *Unanswerable questions* - in other words *Paragraphs with no answers*. In the setting of open domain question answering, not always there is an answer present in the paragraph. To address this issue the SQUAD 2.0 dataset (Rajpurkar, Jia, et al., 2018) was released which contains some questions which do not have an answer.

2. *Discrete reasoning over paragraphs* - To address the simplicity of the task, questions which do not need any symbolic reasoning, (Dua et al., 2019) released a dataset named DROP (Discrete Reasoning Over Paragraphs) which consists of questions which will need to resolve multiple references in a question, perhaps to multiple input positions, and perform discrete operations over them (such as addition, counting, or sorting).

3. *Multihop reasoning* - When a question needs several paragraphs or sentences to find references to entities which are needed to answer a question, the task is referred to as Multihop QA. For instance, a paragraph which contains an answer might not have any lexical term match with the question terms. Such situation makes the Reading Comprehension models to fail (predict wrong answers). Multihop QA models must take this into account by cross referencing entities from different paragraphs and making links to the answer. Figure 2.2 shows an example by (Zhilin Yang, Qi, et al., 2018) where the answer has 5 supporting sentences from 2 different paragraphs which relates the answer to the question.

There are some simple approaches used for the above cases. For *unanswerable questions*, a binary classifier can be trained to determine if a paragraph contains answers to a question, if the binary predication is positive, then the *Start* and *End* linear classifiers can extract the answer from the paragraph. For *Discrete reasoning over paragraphs*, authors of (Dua et al., 2019) propose multi-class classification approach to determine what type of reasoning problem the question refers to and perform further steps. We discuss more works on these topics in the Section 3.

2.2.5 Open Domain Question Answering - OpenQA

Back to where it all started - the pioneering QA task as explained in Section 2.2 is open domain question answering. Open Domain Question Answering or *OpenQA* is a QA task where a given input data would contain a question and a short answer (which is a factoid answer) and no supporting paragraphs or documents. The systems should find relevant documents from some sources, and further find relevant paragraphs and perform answer processing to extract a short answer. In other words, *OpenQA* task follows the QA pipeline, where a QA system as shown in Figure 2.1 shall perform intermediate steps in order to obtain a final answer.

The difference between *Legacy OpenQA* and *OpenQA* is in the intermediate modules. Earlier, different information retrieval approaches, knowledge graphs and ontologies were used to process information in the intermediate modules. Currently deep learning models have replaced these intermediate modules (by modelling different intermediate tasks) and the goal is to achieve an end-to-end model which does all of it in a single model like in multi task learning.

OpenQA task is a union of both *Answer Sentence Selection* and *Reading Comprehension* tasks done sequentially but with certain changes. The assumption of *Reading Comprehension* that the paragraph always contains an answer will not be feasible in *OpenQA* setting as the *Answer Sentence Selection* model might select a paragraph semantically nearby to the question terms but might not contain an answer.

Q: **What's the capital of Ireland?**
P_1: **As the capital of Ireland, Dublin is...**
P_2: **Ireland is an island in the North Atlantic...**
P_3: **Dublin is the capital of Ireland. Besides, Ottawa is one of famous tourist cities in Ireland and ...**

An example from a dataset on *OpenQA* task.

An example shown above shows how a relevant paragraph does not always contain an answer. P_2 is a relevant paragraph according to an Information Retriever model but the answer is not present in this paragraph. A model trained on *Reading Comprehension* task will extract some answer which is incorrect. To overcome this problem, *RC 2.0* models can be used in combination with *Answer Sentence Selection* models or probabilistic models can be used to extract answers only from highest probable paragraph using a *Reading Comprehension* model.

2.3 Datasets/Corpora

The performance of a QA model can be tested and reported on benchmark datasets which are used by the research community. In this section we list some of the well known datasets used for reporting results on particular tasks. As explained in QA pipeline in the Figure 2.1, the section 2.2 details three main tasks and the recent variants of the *Reading Comprehension* task. Listed below are the datasets corresponding to these tasks.

2.3.1 Answer Sentence Selection

Domain	Dataset	Split	#Question	#QA Pairs
Open	Trec QA	Train	94	4,718
		Train-all	1229	53,417
		Raw-Dev	82	1,148
		Raw-Test	100	1,517
		Clean-Dev	65	1,117
		Clean-Test	68	1,442
Open	SQUAD-Sent	Train	87,599	359,222
		Dev	10,570	90,117
		Test	-	-
Open	Wiki QA	Train	873	8,627
		Dev	126	1,130
		Test	243	2,351
Closed	Insurance QA	Train	12887	-
		Dev	1000	-
		Test1	1800	-
		Test2	1800	-

Tab. 2.1: Answer Sentence Selection Datasets. #Questions - Number of questions. #QA Pairs - Number of Question-Answer pairs. Statistics presented by (Lai et al., 2018). WikiQA questions without any correct answers are removed as done by (Y. Yang et al., 2015)

Table 2.1 presents the datasets for the *Answer Sentence Selection* task.

TrecQA is a well known dataset to benchmark a system built for *Answer Sentence Selection* task. It was created from the TREC Question Answering tracks (M. Wang, Smith, et al., 2007). It contains real questions created from search engine logs and sentences from news articles returned by the participating systems in TRECQA task. There are two versions of this dataset, both have the same training set (Train has manual judgements and Train-all is a noisy set with automatic judgements) but their development and test sets differ. The clean version has removed questions in the dev and test sets that did not have answers or only contained positive/negative answers, reducing the development and test sets sizes from the raw version. WikiQA dataset (Y. Yang et al., 2015) is constructed from real queries of Bing search engine and Wikipedia data. The answers are annotated by human annotators on Amazon Mturk crowd sourcing platform. Questions with no correct answers are usually removed, resulting in the statistics presented in the table 2.1.

InsuranceQA by (Feng et al., 2015) is a large scale dataset on insurance domain. The questions and answers were collected from the website Insurance Library[7]. The questions are from real world users and the answers with high quality were composed by professionals with deep domain knowledge. There could be multiple correct answers for some questions so that the number of correct answers is larger than the number of questions.

SQUAD-Sent dataset was created by us by modifying the SQUAD dataset (Rajpurkar, J. Zhang, et al., 2016) designed for reading comprehension, into an answer sentence selection dataset to provide the answers in their original context. Sentence tokenization was performed on SQUAD dataset paragraphs using spacy toolkit[8] and answers were checked for an exact match of strings in the sentences. There is only one positive sentence per question and the all other sentences are negative examples.

Although Natural Questions dataset by (Kwiatkowski et al., 2019) contains long sentence answers (like in sentence selection task) for questions, with both relevant and irrelevant paragraphs. It is not explicitly used for *Answer Sentence Selection* rather it is used for *RC 2.0* or *OpenQA*. Questions come from Google search engine queries filtered by human annotators along with the answers.

2.3.2 Answer Extraction

The Table 2.2 presents different answer extraction task datasets for *Reading Comprehension*, Advancements to *Reading Comprehension - RC 2.0*, and *OpenQA* tasks.

The rise of deep learning models in the field of natural language processing gave rise to the demand for large scale labelled datasets across different tasks. For reading comprehension, the first large scale synthetic dataset was constructed by (Hermann et al., 2015) who used a large scale news domain corpus and converted that into a cloze style question answering dataset who goal is to find the missing entities for the query, in the paragraph. CNN and Dailymail datasets presented in the Table 2.2 shows this first large scale supervised reading comprehension dataset created synthetically which was inspired by the Cloze style QA.

Following this, a work by (Rajpurkar, J. Zhang, et al., 2016) released the SQUAD dataset which presents a new dataset for Reading Comprehension annotated by human crowdsourced workers. A set of human annotators were asked to read a paragraph and frame questions on it. SQUAD dataset is split into *Train*, *Dev* and *Test* sets. The *Test* dataset is hidden from public access. The only way to evaluate a

[7] https://www.insurancelibrary.com
[8] https://spacy.io

Dataset	Annotation	Split	#Question	Task
CNN	Synthetic	Train	380,298	RC - Cloze
		Dev	3,924	
		Test	3,198	
Dailymail	Synthetic	Train	879,450	RC - Cloze
		Dev	64,835	
		Test	53,182	
SQUAD	Human Annotated	Train	87,599	RC
		Dev	10,570	
		Test	9,533	
SQUAD 2.0	Human Annotated	Train	130,319	RC 2.0
		Dev	11,873	
		Test	8,862	
NewsQA	Human Annotated	Train	107,673	RC 2.0
		Dev	5,988	
		Test	5,971	
HotpotQA	Human Annotated	Train-easy	18,089	RC 2.0 Multi Hop
		Train-medium	56,814	
		Train-hard	15,661	
		Dev	7,405	
		Test-Distractor	7,405	
		Test-fullwiki	7,405	
DROP	Human Annotated	Train	77,409	RC 2.0 Multi Hop
		Dev	9,536	
		Test	9,622	
QUASAR-T	Synthetic	Train	37,012	OpenQA
		Dev	3,000	
		Test	3,000	
SearchQA	Synthetic	Train	99,811	OpenQA
		Dev	13,893	
		Test	27,247	
TriviaQA	Synthetic	Train	66,828	OpenQA
		Dev	11,313	
		Test	10,832	
CuratedTREC	Human Annotated	Train	1,486	OpenQA
		Dev	-	
		Test	694	
Webquestions	Human Annotated	Train	3,778	OpenQA
		Dev	-	
		Test	2032	
Natural Questions	Human Annotated	Train	307,373	Ans. Sentence Selection+RC 2.0 +OpenQA
		Dev	7,830	
		Test	7,842	
BIOASQ	Human Annotated	Train	685	OpenQA
		Dev	94	
		Test	161	

Tab. 2.2: Reading Comprehension, OpenQA and modified RC 2.0 datasets.

system on the *Test* set, is by submitting the code on a codalab platform. The SQUAD leaderboard[9] then will display the system scores. A lot of work has been published by benchmarking on this dataset.

[9] https://rajpurkar.github.io/SQuAD-explorer/

Due to the issues pointed out in the Section 2.2.4 with respect to the SQUAD v1.0 dataset on Reading Comprehension, all the other datasets published with changes are referred as *RC 2.0* in the Table 2.2 including SQUAD v2.0.

SQUAD 2.0 dataset (Rajpurkar, Jia, et al., 2018) was released two years after the first one with additional data which contained questions without an answer. The dataset contained an additional data point about being answerable or not. The unanswerable questions were annotated by crowd sourced workers.

NewsQA is a *RC 2.0* type dataset with unanswerable questions. Only the CNN articles from the dataset of (Hermann et al., 2015) were used and crowd sourced workers were asked to create questions in natural language. This is done to convert a synthetic cloze QA dataset to a crowd sourced human annotated QA dataset.

HotpotQA (Zhilin Yang, Qi, et al., 2018) is a special *RC 2.0* dataset which pointed out the issues with *Reading Comprehension* or traditional *RC 2.0* datasets with unanswerable questions being simple to answer without any reasoning required. Another challenge with models trained on these datasets is the interpretability or explanation. Therefore this dataset provides supporting facts from different parts of a paragraph which are required to answer a particular question. This dataset is a *Multi-hop* reasoning dataset which means an answer can be answered with information taken from more than one document to arrive at the answer. This solves another shortcoming of the *RC* datasets such as SQUAD whose questions have high lexical similarity around the answer terms in the paragraphs.

DROP dataset (Dua et al., 2019) is a crowd sourced dataset which is also a *Multi-hop* reasoning dataset like HotpotQA but has questions which require discrete operations such as addition, counting, or sorting to answer questions. Some of the other operations needed to answer these questions involve - subtraction, comparison, selection, addition, coreference resolution, different answer spans and more. The authors also propose a multiclass classifier method to classify which kind of reasoning problem the question would belong to and perform discrete operations.

QUASAR-T dataset (Dhingra, Danish, et al., 2018) consists of trivia questions which were collected manually by a reddit user and posted freely online[10]. It is in the format of *OpenQA* task where the paragraphs are retrieved using LUCENE tool and not all the paragraphs contain the answer.

[10] https://www.reddit.com/r/trivia/comments/3wzpvt/free_database_of_50000_trivia_questions/

SearchQA (Dunn et al., 2017) is a large-scale open domain QA dataset which consists of QA pairs crawled from J![11] archive and the paragraphs are obtained by retrieving 50 webpages for each question from Google search.

TriviaQA dataset (Joshi et al., 2017) consists of QA pairs created by trivia enthusiasts and documents gathered by retrieving 50 webpages per question using Bing Web search. For our experiments involving Quasar-T and SearchQA, we use the retreived paragraphs from (S. Wang, M. Yu, Guo, et al., 2018) as done by (Y. Lin et al., 2018).

CuratedTREC dataset (Voorhees, 2001) is based on the benchmark from the TREC QA datasets from 1999, 2000, 2001 and 2002.

WebQuestions (Berant et al., 2013) is designed for answering questions from the freebased knowledge base which was built by crawling Google suggest API and the paragraphs were retrieved from English wikipedia.

Natural Questions by (Kwiatkowski et al., 2019) is a large scale QA dataset with questions coming from Google search engine which are anonymous and aggregated. They also release a 5 way annotated test data to capture different ways of human annotations. In the paper they report some findings on 25-way annotations on 302 examples to highlight human variability for annotations. This dataset is 3 times bigger than SQUAD dataset and contains long answers (paragraphs or wikipedia HTML bounding boxes highlighted by the annotators) and short answers (answer spans highlighted by the annotators).

There have been some debate about the inductive bias in some of the datasets like *SQUAD* dataset where people were first shown the paragraphs and answers while annotating and were asked to frame questions based on that. This arguably gives an easier set of questions than when somebody was just asked to find answers in a paragraph which the *SQUAD* task is aimed for. Datasets like **Natural Questions** do not ask users to formulate queries but rather choose the queries based on existing query logs from their search engines to avoid such biases.

The field of Question Answering has seen a lot of datasets because of the rise in the usage of deep learning methods and the trend of releasing more models and datasets. This has changed and improved some of the issues in the former QA systems which heavily relied on feature engineering. In the section 3, we discuss some works which have had significant impact in the QA field.

[11] https://www.j-archive.com

BIOASQ dataset

BIOASQ[12] challenge is a large-scale biomedical semantic indexing and question answering task (Tsatsaronis et al., 2015) which has been held successfully for 7 years. The challenge proposes several tasks using biomedical data. One of the tasks focuses on Biomedical question answering (Task B Phase B) where the goal is to extract answers for a given question.

Since our work is mainly focused on domain specific QA with biomedical data, we present the *BIOASQ* data separately. *BIOASQ* task is an *OpenQA* task with both relevant and irrelevant snippets in the dataset.

Datasets	Train	Dev	Test
BIOASQ 4b	427	59	161
BIOASQ 5b	544	75	150
BIOASQ 6b	685	94	161

Tab. 2.3: BIOASQ datasets used in all our experiments along with their splits. The numbers represent number of questions. It is a small scale expert annotated QA dataset.

2.3.3 Open Domain vs Closed Domain Corpus

Closed Domain corpora are those datasets that come from a specific domain source which consists of special definitions, vocabulary terms and features which corresponding to the domain such as Insurance, Medical, Biomedical, Scientific, law etc. Their size is often small scale.

Open Domain corpora are those datasets that come from a broad range of data sources and are not constrained by some special domain specific content. Usually datasets coming from News, Wikipedia, Common crawled data from the internet that do not require certain domain expertise to understand. They are termed under Open Domain data for textual corpora. Their size is often large scale.

BIOASQ Question Answering dataset is a biomedical domain dataset with questions annotated by biomedical domain experts. Domain specific (closed domain) corpus often have certain special characteristics which make them different. Some of the characteristics are listed below:

- Specialized vocabulary.

[12] http://bioasq.org/

- Different answer variants (abbreviations, symbols etc.) for a same question in factoid QA.

- Small scale data.

- Created with the help of domain experts.

- Focused towards a specific community of users.

BIOASQ dataset being a closed domain dataset, was annotated by biomedical experts with the support of biomedical documents. Experts were asked to curate questions and annotate or provide answers to them.

2.4 Evaluation metrics

Different Question Answering tasks have different evaluation metrics because of different outputs. We discuss the evaluation metrics we used in our work in this section based on the corresponding QA task. We experiment mainly on the tasks of *Answer Sentence Selection*, *Reading Comprehension*, *BIOASQ Question Answering* and *OpenQA*.

2.4.1 Answer Sentence Selection

For a given question, there are a set of sentences out of which one or more sentences are correct. For evaluation, Mean Average Precision (MAP) (Eq. 2.1) and Mean Reciprocal Rank (MRR) (Eq. 2.2) are used.

Precision is calculated for each question if the highest ranked (or scored) sentence is retrieved correctly or not (P = 0 for each correct sentence that was not retrieved). The average is then calcuated for each question. Finally, an average over all questions is calculated.

$$MAP = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{Q_j}\sum_{i=1}^{Q_j}P(rel=i) \tag{2.1}$$

with Q_j being the number of relevant sentences for question j; N the number of questions, and $P(rel = i)$ the precision at i^{th} relevant sentence.

In the case of MRR, each system returns some K number of answers, in rank of confidence of their correctness, for each question. The MRR is defined as the mean of the inverse rank of the first correct answer, taken over all N questions:

$$MRR = \frac{1}{N}\sum_{i=1}^{N} RR_i \tag{2.2}$$

$$RR_i = \frac{1}{r_i} \tag{2.3}$$

The score for an individual question i is the reciprocal rank r_i where the first correct answer appeared (0 if no correct answer in top K (often K=5) answers). Thus, there are only six possible reciprocal ranks per question for K=5: 0, 0.2, 0.25, 0.33, 0.5, and 1. MRR is averaged over the N questions.

Instead of implementing code for computing this, once can use the scripts[13] provided by Trec.

2.4.2 Reading Comprehension

For a simple *Reading Comprehension* system, an answer is present inside the paragraph. i.e the answer is a substring of a paragraph string. There are two types of evaluations commonly done by systems for this task.

- Unofficial Evaluation - Word offsets of the answers in paragraphs are evaluated.
- Official Evaluation - As defined by (Rajpurkar, J. Zhang, et al., 2016), answer strings are extracted from the word offsets, and are normalized before being evaluated.

A simple *Reading Comprehension* system has two classifier outputs which detect the start and end positions of the answers in the paragraph.

For the *Unofficial Evaluation* - Accuracy is evaluated to check how accurate the start and end positions individually are extracted by the model. For a question i, if Gold standard label is the same as Predicted label, Accuracy is 1, else the Accuracy is 0 where label is the pair of word offsets. An average over all the questions is calculated.

[13] https://github.com/usnistgov/trec_eval

For the *Official Evaluation* - The predicted word offsets are used to retrieve the answer text from the paragraph, and are normalized to remove punctuations, stop words, extra spaces etc. and are checked for exact match between the gold standard and normalized prediction strings. If both the strings match, then *Exact Match (EM)* score is 1 for question i, else *Exact Match (EM)* is 0, an average over all the questions is calculated.

Along with Exact Match, F1 Score for each question is calculated and averaged over all the questions, using the following equation:

$$F_1 = \left(\frac{2}{\text{recall}^{\,-1} + \text{precision}^{\,-1}} \right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.4}$$

Precision and Recall are computed based on words which are retrieved and are computed as follows:

$$Precision = \frac{len(tokens(prediction) \ \cap \ tokens(gold\,standard))}{len(prediction)} \tag{2.5}$$

$$Recall = \frac{len(tokens(prediction) \ \cap \ tokens(gold\,standard))}{len(gold\,standard)} \tag{2.6}$$

Where $tokens$ function tokenizes words from the string and len function calculates the number of words in the string.

These measures were initially used by (Rajpurkar, J. Zhang, et al., 2016) on the SQUAD leaderboard[14] whose code can be found in their repository.

2.4.3 BIOASQ evaluation

BIOASQ challenge has a biomedical question answering (Task B Phase B) task where the goal is to extract answers for a given question from relevant snippets.

For a given question, there are one or more relevant snippets which contain the answer. Unlike *Reading Comprehension* task, not all answers are correctly annotated in the gold standard data and also not all gold standard answers are present in all the relevant snippets.

The official evaluation as defined by the task organizers is as follows. There are three measures computed by the scripts[15] provided.

[14] https://rajpurkar.github.io/SQuAD-explorer/

[15] https://github.com/BioASQ/Evaluation-Measures

- Strict Evaluation - Top 1 accuracy - Where accuracy is measured using Exact Match (EM) as done by *Reading Comprehension* tasks for Top-1 answer.

- Lenient Evaluation - Top 5 accuracy - If the gold standard is present in Top 5 responses, where accuracy is measured using Exact Match (EM) as done by *Reading Comprehension* tasks.

- Mean Reciprocal Rank - As done in *Answer Sentence Selection* - the score for an individual question i is the reciprocal rank r_i where the first correct answer appeared (0 if no correct answer in top five answers).

All the three measures are averaged over total number of questions and are reported in the challenge leader board[16], batch wise.

2.5 Contributions and Conclusion

Our work is funded by the GOASQ project[17] which intends to investigate, compare, and combine two different approaches for answering questions formulated in natural language over textual, semi-structured, and structured data. One approach is the text-based question answering that directly answers natural language questions using natural language processing and information extraction techniques. The other tries to translate the natural language questions into formal, database-like queries and then answer these formal queries w.r.t. a domain-dependent ontology using database techniques. This thesis work is focused on the first approach that directly answers natural language questions using natural language processing and information extraction techniques.

As defined in our research objectives (Section 1.3), our focus is towards building QA models which work with better performance both on large scale and small scale datasets such as biomedical domain datasets. For this purpose, we show how *Domain Adaptation* can be done on biomedical data from open domain QA datasets using *Reading Comprehension*. We evaluate two models (Reading Comprehension and OpenQA) for biomedical question answering and compare their performance variation to show that the *OpenQA* modelling of BIOASQ performs better. We also experiment with several open domain datasets for *Domain Adaptation* process to show which ones perform better while adapted to biomedical domain.

[16] http://participants-area.bioasq.org/
[17] https://goasq.lri.fr/

We use SQUAD v1.0, v2.0, HotpotQA, NewsQA datasets for reading comprehension, QUASAR-T dataset for OpenQA, BIOASQ dataset for domain specific dataset. WikiQA, TrecQA for Answer sentence selection task.

Another research objective is to focus on leveraging structured and semantic information for existing QA models. In factoid question answering where answers are usually entities, their types (based on a custom taxonomy) can often be inferred by analysing the question keywords. These types are called as "Expected Answer Types" which are useful in traditional pipeline question answering systems. These are used to filter or eliminate candidates which does not belong to the question type. This information is seldom used explicitly in deep learning models for QA as the goal is to build end-to-end systems. We present a detailed analysis on "Expected Answer Types" for biomedical and open domain QA to verify its usefulness. We also show how entity information and expected answer type information can be used for QA tasks in open domain QA to further improve scores of deep learning models.

We annotate for more variants of answers than just use gold standard data on biomedical QA dataset and show that the *Reading Comprehension* models perform better when annotations are done correctly with more information. We also show how these annotations can be automatically annotated using Metamap tool which uses UMLS meta thesaurus and fetches similar results to the manual ones.

Deep learning models almost always focus towards building end-to-end systems, and not much emphasis is put towards post processing of the outputs to better rank the Top-K predictions. We use semantic features and structured information from different paragraphs provided for a question, and use it for a ranking model to improve the QA performance. We use some traditional machine learning models to rank a better answer candidate from Top-K predictions into Top-1 position and show that there is a scope for improvement on the predictions from the neural network models. This applies both on biomedical question answering and open domain question answering models on *Open QA* tasks.

The following chapter presents some of the state-of-the-art models on the above tasks in detail.

3 State of the Art

In the previous chapter we have defined the Question Answering (QA) tasks, datasets, evaluation metrics commonly used. In this chapter we present the state of the art models and present a literature review of how some models and tasks evolved over time.

We present the history of question answering system development from their initial stages, and how they ended up with a structured pipeline of tasks which we see these days. The extensive use of deep learning models in machine learning also had an impact on question answering changing the way some traditional systems work. The following sections first present question answering in the past, before deep learning and with deep learning models. Recently since contextual embeddings using pre-trained language models have started to improve performance of several NLP tasks including question answering, we present some of the works in this regard which are currently in the trend.

Since a long time, Question Answering is always targetted in different ways based on the type of data used. One of the pioneering works in the field of Question Answering was BASEBALL (Green Jr et al., 1961) which was built for answering questions about baseball games played in the American league over one season. LUNAR (Woods, 1973) was designed as a result of the Apollo moon mission, to help lunar geologists. Several others early systems SYNTHEX, LIFER, and PLANES are mentioned by (Paris, 1985) for the same objective of getting an answer for a question asked in natural language. There were no pipeline structure approaches as we see in Figure 1.1 earlier in those systems and people built rule based systems generally. This kind of QA systems were also built for querying databases. The QA tasks on plain text with answer sentences and spans, gained attention while TREC QA task was organized in 1999. Data such as databases, knowledge bases and graphs, triples, plain text etc. lead to different types of question answering systems. Given a question in natural language, one of the main challenges is to convert or translate the question into a query which can be used to querying a database or a knowledge base. The process involves aligning a question with the KB triples, which needs to overcome lexical gap and to adapt the question parsing to the KB schema in order to determine which phrases are entity or relation mentions. If the question answering data is only based on free text, these problems do not exist but other problems such as text understanding, part of speech tagging, entity recognition, answer type detection etc. are important to tackle (Grau and Ligozat, 2018)

Our work focuses on text based question answering, mainly factoid question answering and answer sentence selection tasks.

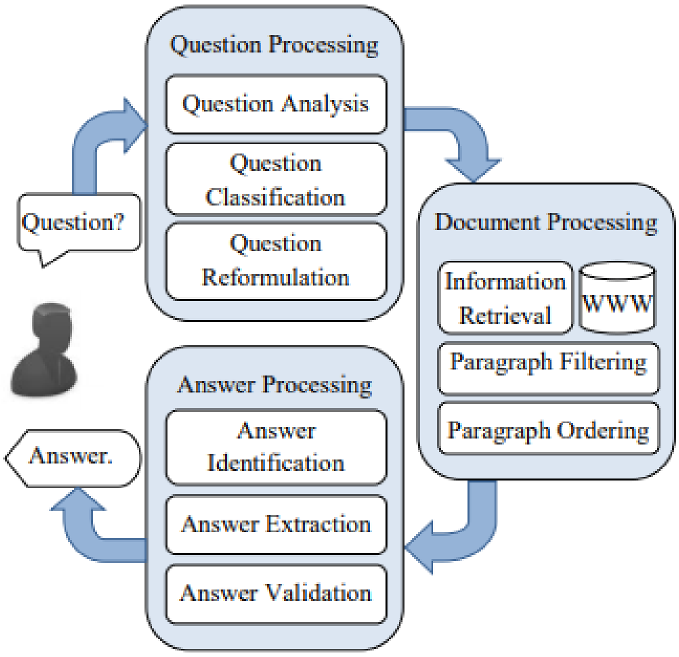


Fig. 3.1: Question Answering pipeline as defined by (Allam and Haggag, 2012)

3.1 Text Based Question Answering by Feature Engineering Approaches

Different question answering systems can be generalised into a single structured pipeline manner. A recent survey by (Allam and Haggag, 2012) shows in Figure 3.1 how different sub tasks are structured into one QA structure and each contribution would come under some sub tasks generally, which is similar to the one presented in Figure 2.1. We list and explain some of the works with respect to the structured pipeline.

To retrieve an answer in a text from a question, there are three main modules namely:

- Question Processing

- Document Processing
- Answer Processing

Question Processing

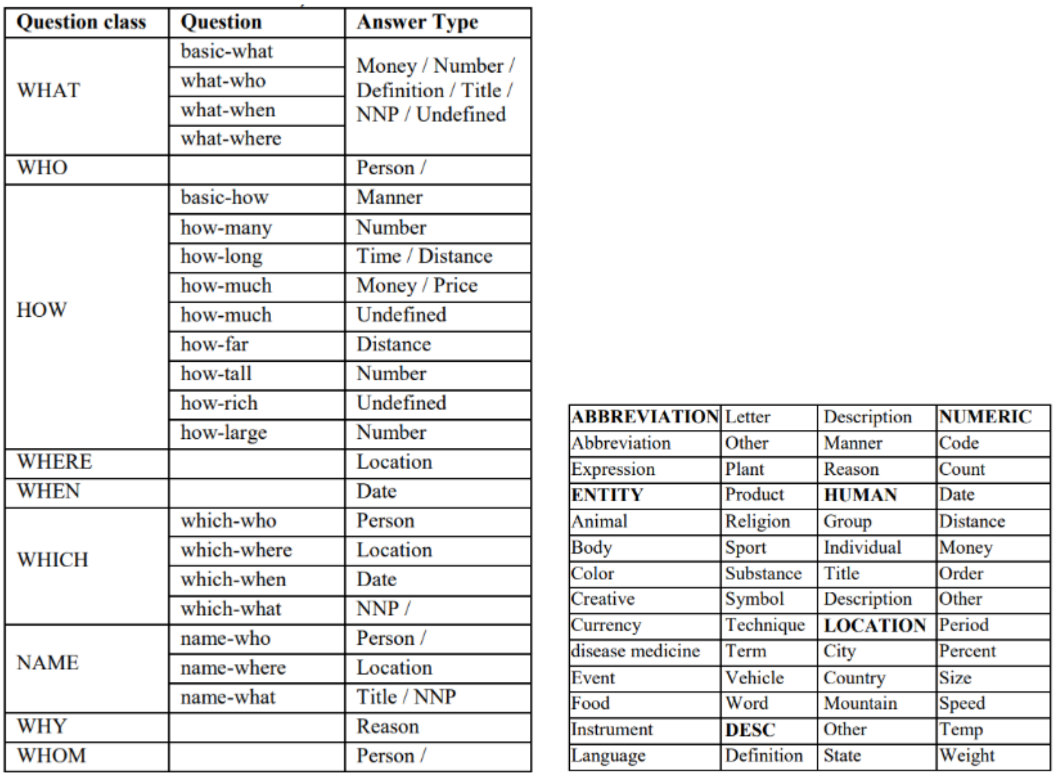
Question Processing module takes as input a question in natural language. This module *Analyzes* the question, *Classifies* it into a question type and *Reformulates* into queries for the IR engine.

Question analysis is done to analyze and determine the *focus* of the question. A *focus* is word or a sequence of words in the question which is used to determine and disambiguate what the question is looking for. For example, in the question "What is the longest river in France?", the question *focus* would be "longest river". If the Expected Answer Type (that defines the entity type of the answer inferred from the question) and the question *focus* are known, the system can determine the answer. Pattern matching rules can be used to determine the *focus* based on the question type. Machine learning methods using models like Wapiti by (Lavergne et al., 2010) that uses conditional random fields for sequence labelling tasks can also be used for this purpose.

Determining the question type is important to understand what kind of answer is to be extracted. This information will make the search easier by filtering the type of information to be searched. For example, if the question type is a "location" then the answer expected shall be a location and not a name of a person. Either done by rule based systems or machine learning based classifiers, the question types are classified based on taxonomies.

The effectiveness of the taxonomy chosen is directly linked to the capacity of recognizing the question type. For rule based systems, the taxonomies act as a base to construct rules such as "Where" questions would classify as "Location" class and "When" would classify as "Date" class and so on. This approach was used by (Moldovan et al., 1999; Hermjakob, 2001; Radev et al., 2002; Ferret et al., 2001) as it was simple, quick and effective.

Machine learning classifiers need labeled questions to be trained on. Choosing the right set of features to represent the question and the type of classifier plays an



Question class	Question	Answer Type
WHAT	basic-what	Money / Number / Definition / Title / NNP / Undefined
	what-who	
	what-when	
	what-where	
WHO		Person /
HOW	basic-how	Manner
	how-many	Number
	how-long	Time / Distance
	how-much	Money / Price
	how-much	Undefined
	how-far	Distance
	how-tall	Number
	how-rich	Undefined
	how-large	Number
WHERE		Location
WHEN		Date
WHICH	which-who	Person
	which-where	Location
	which-when	Date
	which-what	NNP /
NAME	name-who	Person /
	name-where	Location
	name-what	Title / NNP
WHY		Reason
WHOM		Person /

ABBREVIATION	Letter	Description	**NUMERIC**
Abbreviation	Other	Manner	Code
Expression	Plant	Reason	Count
ENTITY	Product	**HUMAN**	Date
Animal	Religion	Group	Distance
Body	Sport	Individual	Money
Color	Substance	Title	Order
Creative	Symbol	Description	Other
Currency	Technique	**LOCATION**	Period
disease medicine	Term	City	Percent
Event	Vehicle	Country	Size
Food	Word	Mountain	Speed
Instrument	**DESC**	Other	Temp
Language	Definition	State	Weight

Fig. 3.2: Hierarchical question types taxonomy by (Moldovan et al., 1999) (left) and (Li and Roth, 2002) (right)

important role on the performance. Features may vary from simple shallow word analysis to detailed syntactic and semantic features using linguistics analysis. The authors of (Hermjakob, 2001) used machine learning based parsing and question classification. The authors of (D. Zhang and W. S. Lee, 2003) compared various choices for machine learning classifiers using the hierarchical taxonomy proposed by (Li and Roth, 2002). Among: Support Vector Machines (SVM), Nearest Neighbors (NN), Naive Bayes (NB), Decision Trees (DT), and Sparse Network of Winnows (SNoW), they showed that with only surface text features SVM outperforms four other methods for question classification.

For the question reformulation into queries (Azad and Deepak, 2019) used entity recognition tools, stop words, part of speech taggers to extract keywords and features from question words which are appended to the expanded query along with synsets from the WordNet by (Miller, 1995). These are input to Information Retrieval engine.

Document Processing

The first step in document processing module is to extract relevant documents for an input question. A document can be defined as a set of multiple paragraphs of text. A relevant document is a document which has some semantic relationship with the question. This can be based on matching terms between question and documents, terms which are semantically nearby to each other measured using cosine similarity in embedding space, etc. An important detail to highlight here is that a relevant document (judged according to some measure) might not always contain short answer terms but might still be relevant to the question, which leads to different QA tasks.

The authors of (Stoyanchev et al., 2008) present a document retrieval model on a question answering system, and evaluate the use of named entities, part of speech tags in a query and show that phrases extracted from questions, named entities of noun, verb, and prepositional phrases improves IR performance than just words. The authors of (Gaizauskas and Humphreys, 2000) describe an IR model with an NLP model that performed reasonably through linguistic analysis.

In question answering datasets which require a relevant textual paragraph for a question, researchers typically use an IR method to retrieve relevant documents. The authors of (Dhingra, Mazaitis, and William W Cohen, 2017a) use ClueWeb09 service to retrieve 100 HTML documents per question and later do post processing to remove HTML syntax, non textual data, images etc. For paragraph filtering and ordering given in the Figure 3.1, different approaches do it differently. A common way of doing this is by splitting documents into paragraphs which act as answer candidates for *Answer Sentence Selection* task which predicts the best suitable paragraphs as answers. We detail this in section 3.2.1. Several models and approaches for this can be found on the ACL leaderboard[1].

Answer Processing

Answer Processing module functions differently for different sub-tasks of textual question answering. For long answers like answer sentences, semantic similarities are captured between questions and answer sentences. For short answers like short spans or phrases, entities, one of the several ways of processing answers is based on the expected answer types obtained in the *Question Processing* module as done by (Grappy, Grau, et al., 2011).

[1] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

For long answers like in TrecQA dataset[2], the authors of (Severyn and Moschitti, 2013) use the answer type and their named entities as features in their tree kernel approach on answer sentence selection and extraction task. The authors of (Gleize and Grau, 2015a) also propose a unified kernal approach for recognizing textual entailment and answer sentence selection tasks.

For short answers, the answer type can be used to infer short answers by using the named entity recognition tools. For the short answers that are entities, a question type classifier would predict a class based on the hierarchies in Figure 3.2, which can be used to extract the matching named entities that are further ranked based on some features such as embedding similarity, term overlap. The authors of (Shih et al., 2005) propose an approach to extract answers in Chinese texts. The authors of (Ravichandran and Hovy, 2002) present a model for finding answers using shallow surface text patterns and manually constructed rules on the Web dataset and TREC-10 questions. The authors of (Peng et al., 2005) present an approach to capture long distance dependencies using linguistic structures to enhance patterns in chinese QA data.

Earlier works before deep learning models such as the ones by (Punyakanok et al., 2004; Cui et al., 2005) focus mainly on syntactic features such as using dependency trees and relation between terms and the distance between question and sentence syntax trees. They also include named entity features for semantic information. Instead of using strict word match between question terms and answer sentence terms, the authors of (Cui et al., 2005) propose fuzzy relation matching based on statistical models. The authors of(Gleize and Grau, 2015a) use kernel functions to detect paraphrases, answer sentence selection and recognizing textual entailment. Some other works such as (Heilman and Smith, 2010; M. Wang and Manning, 2010; Yao et al., 2013b; Yih et al., 2013) focus on using tree edit distances, probabilistic tree edit models, feature extraction using dependency trees, relations, named entity types etc.

3.2 Neural Question Answering - Task based classification

Ever since the rise in the usage of deep learning models and techniques in the field of machine learning and natural language processing. Question answering domain also has witnessed the impact on a lot of question answering systems built using neural network models. For a few sub-tasks of question answering, like answer extraction

[2] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

from plain text, sentence selection task, these models outperform the traditional methods.

From building individual sub-tasks of question answering organized in a pipeline to building end-to-end models which completely rely on input data, deep learning is being used extensively.

Deep learning models are good at automatic feature extraction which the models learn based on the input data, contrary to traditional models which heavily rely on the hand crafted features provided by tools.

In the following section we discuss how deep learning models function on a few QA tasks following the same classification of tasks presented in Section 2. We discuss about the current state of the art models in the respective tasks, how some popular neural network models evolved over time and the advantages/disadvantages of using one model over the other.

3.2.1 Answer Sentence Selection

Given a question and a set of potential answer sentences, answer selection is the task of identifying which of the candidates sentences answer the question correctly. As per the question answering pipeline described in Figure 3.1, the *Document Processing* module outputs a list of paragraphs corresponding to a question, and the *Answer Processing* module shall predict which of the answer sentences (or paragraphs) are relevant to answer the question.

For this task, there are several popular labelled datasets used to benchmark results as listed in Table 2.1. The most popular dataset used by the community since more than a decade is TREC QA dataset curated and first used by (M. Wang, Smith, et al., 2007) where the authors use probabilistic quasi-synchronous grammar for question answering. The dataset is designed for open domain question QA task.

Few years later, (Y. Yang et al., 2015) released a dataset named WikiQA based on Bing search engine query logs and answer sentences were annotated by humans on a crowdsourcing platform. Until this dataset was released, all the models were experimented on TrecQA dataset alone. This shows the lack of datasets for question answering and especially for the task of answer sentence selection. Table 2.1 presents the four datasets out of which two (TrecQA and WikiQA) are widely used for benchmarking models. This task also corresponds to a textual entailment task. Often the models are also evaluated on a paraphrase corpus (Gleize and Grau, 2015a).

Models and approaches

The ACL page[3] for state of the art models for question answering contains a leaderboard with scores comparing MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank) of different models evaluated using official evaluation scripts[4] of Trec. Authors can upload their scores directly on the leaderboard by linking their published articles.

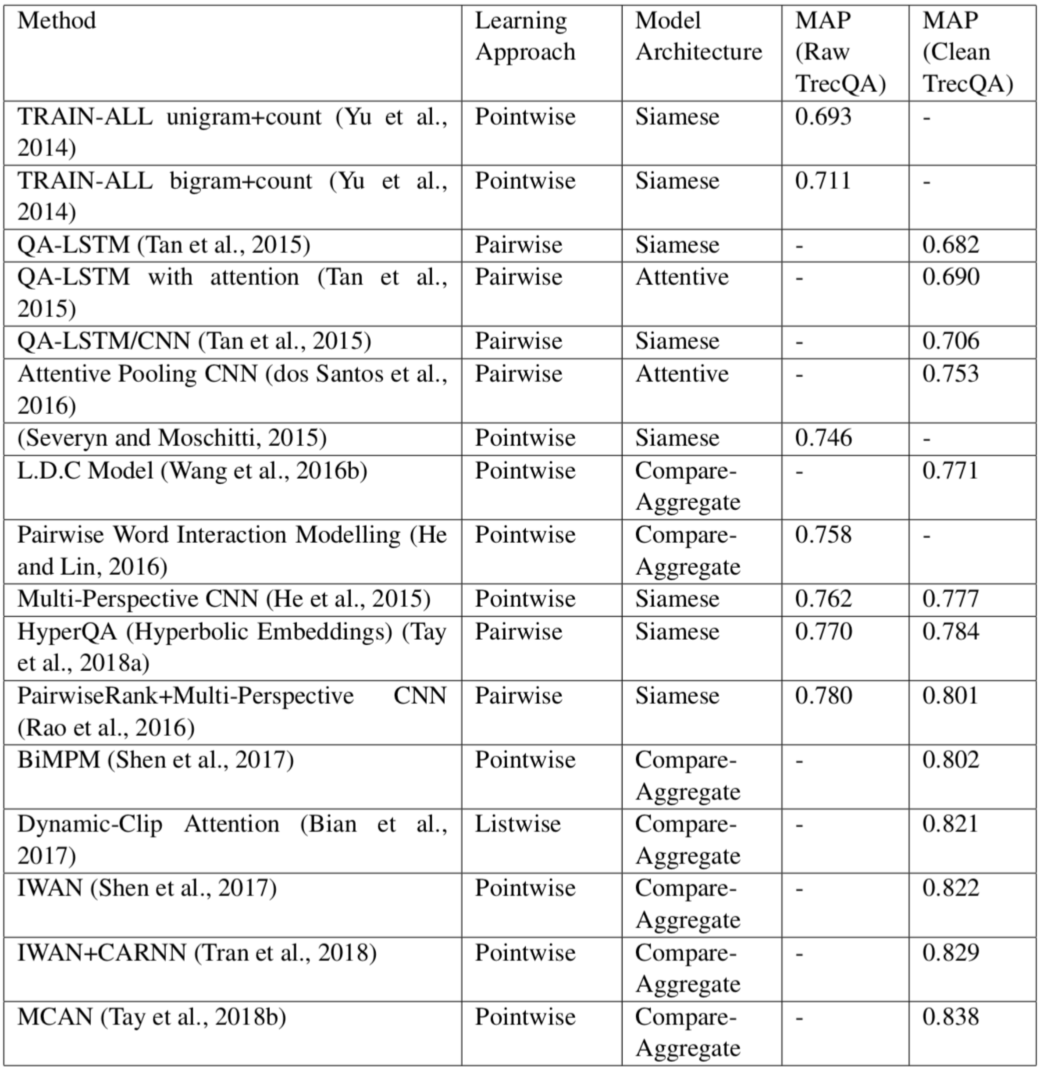
As soon as a deep learning model was first used in this task by (L. Yu et al., 2014) where the authors use a 1D CNN model, others started using different deep learning models for this task. Following the convolutional neural network approach, (Severyn and Moschitti, 2015) propose a siamese CNN model using learning to rank approach, which computes a representation of both entries, candidate passage and question, and a similarity between these two representations using a pooling layer followed by similarity matrix computation. In (Yin et al., 2016), the similarity of the two entries is evaluated by computing interactions between words of the two texts by an attention layer.

The authors of (He et al., 2015) propose a Multi-Perspective CNN for this task which is further used by (Rao et al., 2016) with a triplet ranking loss function to learn pairwise ranking from both positive and negative samples. CNNs are generally used for classification problems where the input size between question and answer pair does not vary significantly. (Lai et al., 2018) gives a good comprehensive review of the task and summarizes several works which use deep learning models.

According to the authors of (Lai et al., 2018), there are three general architectures for measuring the relevance of a candidate answer sentence to a question.

1. **Siamese Architecture**: In a siamese architecture (Bromley et al., 1994), The same encoder (a CNN or a RNN) layer is used to build the representations for the input sentences (both the question and the answer sentence) individually. After that, the relevance score is determined based on the encoded representations. There is no explicit interaction between the input sentences during the encoding process.

[3] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)
[4] https://github.com/usnistgov/trec_eval



Method	Learning Approach	Model Architecture	MAP (Raw TrecQA)	MAP (Clean TrecQA)
TRAIN-ALL unigram+count (Yu et al., 2014)	Pointwise	Siamese	0.693	-
TRAIN-ALL bigram+count (Yu et al., 2014)	Pointwise	Siamese	0.711	-
QA-LSTM (Tan et al., 2015)	Pairwise	Siamese	-	0.682
QA-LSTM with attention (Tan et al., 2015)	Pairwise	Attentive	-	0.690
QA-LSTM/CNN (Tan et al., 2015)	Pairwise	Siamese	-	0.706
Attentive Pooling CNN (dos Santos et al., 2016)	Pairwise	Attentive	-	0.753
(Severyn and Moschitti, 2015)	Pointwise	Siamese	0.746	-
L.D.C Model (Wang et al., 2016b)	Pointwise	Compare-Aggregate	-	0.771
Pairwise Word Interaction Modelling (He and Lin, 2016)	Pointwise	Compare-Aggregate	0.758	-
Multi-Perspective CNN (He et al., 2015)	Pointwise	Siamese	0.762	0.777
HyperQA (Hyperbolic Embeddings) (Tay et al., 2018a)	Pairwise	Siamese	0.770	0.784
PairwiseRank+Multi-Perspective CNN (Rao et al., 2016)	Pairwise	Siamese	0.780	0.801
BiMPM (Shen et al., 2017)	Pointwise	Compare-Aggregate	-	0.802
Dynamic-Clip Attention (Bian et al., 2017)	Listwise	Compare-Aggregate	-	0.821
IWAN (Shen et al., 2017)	Pointwise	Compare-Aggregate	-	0.822
IWAN+CARNN (Tran et al., 2018)	Pointwise	Compare-Aggregate	-	0.829
MCAN (Tay et al., 2018b)	Pointwise	Compare-Aggregate	-	0.838

Fig. 3.3: An overview of deep learning methods applied for *Answer Sentence Selection* presented by (Lai et al., 2018)

2. **Attentive Architecture**: Rather than encoding representations independently, attention mechanisms can be used to allow the information from an input sentence to influence the computation of the other's representation (Tan et al., 2015; Santos et al., 2016).

3. **Compare-Aggregate Architecture**: In a Compare-Aggregate architecture, vector representations of small units such as words of the sentences are first compared. After that, these comparison results are aggregated to calculate the final relevance score.

The authors of (Lai et al., 2018) also note that boundaries of defining which model follows which architecture is often unclear because some models mix different representations inspired from different types of models making it harder to classify under a single category.

Following these definitions for neural model architectures and definitions for *Pointwise* and *Pairwise* learning approaches as defined in Section 2.2.2, the Figure 3.3 presents an overview of deep learning models used for this task. A similar overview can also be found on the ACL leaderboard[5].

The authors of (Tayyar Madabushi et al., 2018) use a CNN model proposed by (He et al., 2015) using question classes (or question types as described earlier) to enhance the dataset by highlighting entities in it. Highlighting entities were done by mainly two ways called Bracketing (appending a special token before and after the entity occurrence) and Replacement (replacing the entity word with a special token) methods. They propose the above two methods for highlighting expected answer types or question classes in the answer text.

Answer Sentence Selection task was extensively studied and several works using different neural architectures were proposed. But the question answering community shifted focus towards tougher aspects of the factoid QA tasks such as answer extraction and answering tougher questions which require reasoning skills than just factoid answer type matches.

3.2.2 Reading Comprehension

Reading Comprehension task has been addressed in several ways. One of the popular tasks in CLEF was *QA4MRE* by (Peñas et al., 2013) which provided multiple choice questions (with one correct answer) and the goal was to understand single documents and answer a question out of the options provided.

Two of the other earliest Reading Comprehension systems are based on pattern matching techniques with bag-of-words (Hirschman et al., 1999), and a rule based system *Quarc* (Riloff and Thelen, 2000) which use rules based on the question words present in the question. The dataset consist of 115 questions in total. The authors of (Poon et al., 2010) propose an approach using information extraction methods for detecting predicate argument triples that can later be queried as a relational database, similar to converting queries in natural language to structured queries. The authors of (Gleize and Grau, 2015b) use word vectors and tree edit model on graph representations of the passages and answer choices to extract edit sequences which decide the correct answers among several choices of *QA4MRE* task (Peñas et al., 2013).

[5] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

With the introduction of a new dataset by (Hermann et al., 2015), cloze style QA task was used to convert news domain data into QA data. Since this was synthetic, human annotated data like SQUAD dataset by (Rajpurkar, J. Zhang, et al., 2016) became more popular as they were harder than the synthetic ones. The task is to identify an answer span (a short answer) in the given paragraph which is the correct answer. The models built for this task must focus on analyzing the given paragraph to extract an answer from it. The main assumption of this task is that the answer is always present in the paragraph. This task is often referred as *Machine Reading* as well.

The lack of training data is one of the bottlenecks of using deep learning models. Not just question answering, but this is common to many other NLP tasks. A solution for this bottleneck is to create more data (labelled data). There are three common ways of doing this,

- Creating labelled datasets by automatically generating question and answers from a source (Hermann et al., 2015) - Synthetic datasets.

- Creating semi-supervised models with limited labelled data (Dhingra, Mazaitis, and William W Cohen, 2017a) - Semi-supervised datasets.

- Creating labelled datasets by human annotations (Rajpurkar, J. Zhang, et al., 2016) - Human annotated datasets.

Datasets

Synthetic datasets are the ones that can be created automatically and without expert knowledge for annotations. The authors of (Hermann et al., 2015) followed the approach of Cloze style reading where a summary or a paraphrase sentence related to a paragraph is used to create queries. The answer terms are one or more entities present in the query, but hidden or anonymized. The task is to identify which is the entity hidden in the query. The dataset created and released by the authors contained more than 1 million query answer pairs.

The follow up work by (Chen, Bolton, et al., 2016) criticizes the way these questions are framed and state that the required reasoning and inference level of this dataset is still quite simple. The authors build a simple model which outperforms the state of the art by 7-10% of the original paper results. Although this dataset is relatively simple to address the complex reading comprehension task, this work fetched more attention towards the reading comprehension task especially towards using an end-to-end model.

Addressing the issues discussed above and the noise present in automatically created synthetic datasets, (Rajpurkar, J. Zhang, et al., 2016) released a new human annotated dataset for reading comprehension named as SQUAD (Stanford QUestion Answering Dataset). This dataset contains over 100,000+ questions posed by crowdworkers on wikipedia articles. The workers were asked to read a paragraph and frame questions based on the paragraph and mark the answer span in the paragraph. This was a pioneering work in reading comprehension which released a human annotated large scale dataset which led to several other works in the domain of reading comprehension for question answering.

Along with the dataset, Kaggle[6] like leaderboards for question answering was released[7] by the authors of (Rajpurkar, J. Zhang, et al., 2016) which made research on reading comprehension more competitive and comparable. The authors propose a baseline approach using logistic regression and textual features to extract answers and also report human accuracy score which was computed by comparing one of the three answers humans had annotated as a prediction and others as gold standard.

Although a similar leaderboard approach existed already for *Answer Sentence Selection*[8], the main difference introduced by the authors was a hidden test set. A participant must submit the code and model to the organizers in order to evaluate their model. That made sure that the results were not fine-tuned on the test set and the scores were reproducible and trustworthy which made a leap in transparency of reported results.

A lot of interesting approaches, tricks and models came out as a result since the availability of this dataset which showed how deep learning models would indeed perform better with more data. Most of these below mentioned works can be found on the SQUAD leaderboard[9] as well.

Models and approaches

We briefly discuss some of the works contributing towards *Reading Comprehension* and mainly on SQUAD dataset and highlight some important points from the articles.

[6] https://www.kaggle.com/

[7] https://rajpurkar.github.io/SQuAD-explorer/

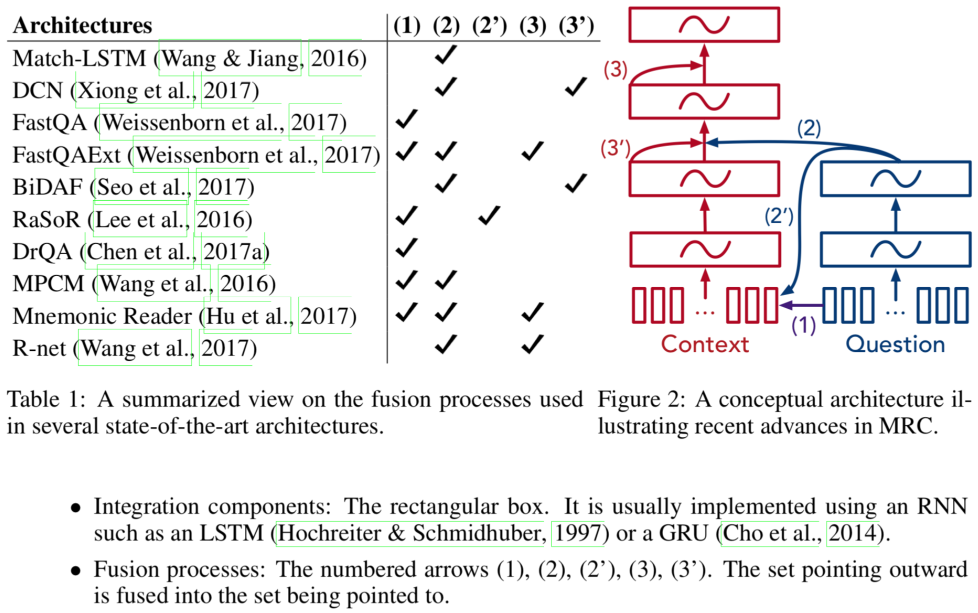
[8] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

[9] https://rajpurkar.github.io/SQuAD-explorer/

When variable length textual data is used such as in paragraph texts where the size might vary significantly from one to another. Using a Convolutional Neural Network (CNN) is not very straightforward as CNNs were modelled on images initially which are fixed size inputs. Several works on *Reading Comprehension* focus on using Recurrent Neural Networks such as Long Short Term Memories (LSTM) or Gated Recurrent Units (GRU) to model sequential and variable length data. Works by (Weissenborn et al., 2017; Chen, Fisch, et al., 2017) use a similar simple architecture of LSTM or GRUs for both Question and Paragraph encoding and use a mechanism to learn interaction between question layer and paragraph layer. The authors of (Weissenborn et al., 2017) call it as *Interaction Layer* and (Chen, Fisch, et al., 2017) call is it as *Aligned Question Embedding* which makes an interaction of question terms with paragraphs term possible. The authors of (S. Wang and Jiang, 2016) used MatchLSTM which is built on Pointer networks by (Vinyals et al., 2015) where the output sequence tokens must come from the input sequence. Instead of picking an output token from a fixed vocabulary, pointer network uses attention mechanism as a pointer to select a position from the input sequence as output. This was one of the first few models which used attention mechanism for this task.

In the same regard, attention mechanism which was also used extensively in sequence to sequence models for machine translation and later modified to work on Reading Comprehension by several others and almost all models using RNNs use some kind of attention mechanisms for this task to facilitate interaction between two sequences. Using attention in a bidirectional manner was later shown by (Seo et al., 2016) termed as *BIDAF - BIDirectional Attention Flow* which uses a hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity. BIDAF includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation. Inspite of this model being more complex than DRQA by (Chen, Fisch, et al., 2017), the latter performs much better on the SQUAD leaderboard (BIDAF - EM 67.974, DRQA - EM 70.733), which is indeed a surprising aspect.

Ever since RNN models became popularly used for this task, works by (Xiong et al., 2016; Hu et al., 2017; Shen et al., 2017; Huang et al., 2017) focused on using attention mechanism in different ways to handle dependency between question to paragraph, paragraph to question, left to right, right to left etc. by modelling different mechanisms with RNNs. Figure 3.4 by (Huang et al., 2017) gives a comparison between different works by showing which model uses how many RNN, at what level



Architectures	(1)	(2)	(2’)	(3)	(3’)
Match-LSTM (Wang & Jiang, 2016)		✓			
DCN (Xiong et al., 2017)		✓			✓
FastQA (Weissenborn et al., 2017)	✓				
FastQAExt (Weissenborn et al., 2017)	✓	✓		✓	
BiDAF (Seo et al., 2017)		✓			✓
RaSoR (Lee et al., 2016)	✓		✓		
DrQA (Chen et al., 2017a)	✓				
MPCM (Wang et al., 2016)	✓	✓			
Mnemonic Reader (Hu et al., 2017)	✓	✓		✓	
R-net (Wang et al., 2017)		✓		✓	

(3)
(2)
(3’)
(2’)
(1)
Context
Question

Table 1: A summarized view on the fusion processes used in several state-of-the-art architectures.

Figure 2: A conceptual architecture illustrating recent advances in MRC.

- Integration components: The rectangular box. It is usually implemented using an RNN such as an LSTM (Hochreiter & Schmidhuber, 1997) or a GRU (Cho et al., 2014).
- Fusion processes: The numbered arrows (1), (2), (2’), (3), (3’). The set pointing outward is fused into the set being pointed to.

Fig. 3.4: Summary of several models using an attention mechanism for Reading Comprehension. Figure by (Huang et al., 2017).

and how many attentions are used. This comparison shows how little modifications in the neural architectures led to different results.

Although the focus is to increase metrics like exact match score or F1 score of these models for Reading Comprehension, the training time of these models is increased along with model complexity while using RNNs. That is one of the disadvantages of using RNNs (it is worse while multiple RNNs are used sequentially) where the training time is increased proportionally to increasing complexity of the RNN model.

Convolution operations on the other hand take lesser training times and hence they are better suited when time constraint is considered seriously. Following this problem (A. W. Yu et al., 2018) propose *QANET* to use local convolutions and self attention mechanism by (Vaswani et al., 2017) which would avoid the use of RNNs but facilitates using attention mechanism. The self attention mechanism was one of the pioneering works which was famously called as *Transformer model* which later was used for contextual embeddings. Modifications of RNNs had almost hit a ceiling on SQUAD leaderboard followed by *QANET* which performed better than the previous RNN methods whose exact match score for a single model was 82.47%. This is the peak performance reported on SQUAD leaderboard for a model that does not use pre-trained language representations, which are explained in the following section.

Pre-trained language representations and reading comprehension

An important milestone of using pre-trained language representations for contextual word embeddings was proposed by (M. Peters et al., 2018a) in their work *Deep contextualized word representations* which is famously named as *ELMO - Embeddings from Language Models*. The authors train a deep bidirectional language model on a large corpus to learn word vectors, these word vectors are later added as an embedding layer to downstream NLP tasks. The authors concatenate these contextual word vectors into existing state of the art models for NLP tasks such as question answering (Reading comprehension on SQUAD), textual entailment, semantic role labelling, co-reference resolution, sentiment analysis, named entity recognition and show that the performance of these models can be improved right away without modifying anything with the downstream task model. This was originally inspired by TagLM (M. E. Peters et al., 2017) which shows the feasibility of this on sequence labelling tasks.

The authors of *ELMO* (M. Peters et al., 2018a) use a baseline model of theirs which is an improved version of BIDAF model which fetches exact match score of 81.1 and adding *ELMO* words vectors to the same model fetches exact match score of 85.8 which is an increase 4.7 points on the SQUAD dataset test set. This work sets a new baseline for using contextual word embeddings trained on language model tasks.

ELMO uses Bi-LSTMs in their model in both the directions left and right, separately to encode the sequences. Instead of these Bi-LSTMs, (Radford et al., 2018) propose to use a transformer model as proposed by (Vaswani et al., 2017). Following *ELMO* and *GPT*, a work by (Devlin et al., 2018) named as *BERT - Bidirectional Encoder Representations from Transformers* was released. BERT is also a popular name from the Sesame Street[10] which is why a muppet is used while portraying the BERT model. Figure 3.5 compares the three models briefly.

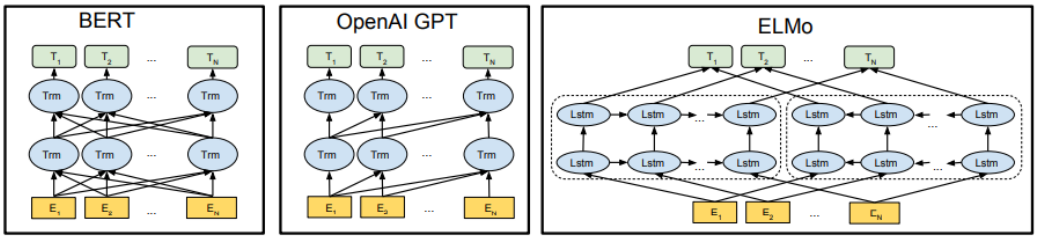


Fig. 3.5: Comparing BERT by (Devlin et al., 2018) with OpenAI GPT (Radford et al., 2018) and ELMO (M. Peters et al., 2018a)

[10] https://www.sesamestreet.org/

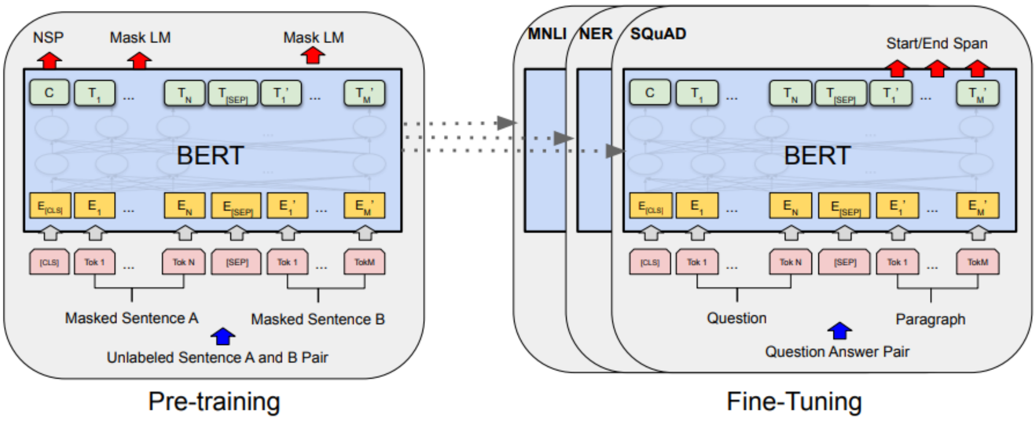


Fig. 3.6: Overall process of pre-training and fine tuning *BERT* for different NLP tasks by (Devlin et al., 2018)

BERT is designed to pretrain deep bidirectional word representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained *BERT* model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks, such as question answering and language inference, without substantial task specific architecture modifications.

For question answering, as shown in Figure 3.6, the final layer of *BERT* is an additional layer which shall predict the *Start* and *End* of the answer span, with no new model specifically for QA. BERT-Large (A variant with large number of layers) scored 84.1% which beat the previous state of the art models at the time of release and the only peer reviewed and published paper at the time of writing. Other variants of BERT and better models have not been published yet althought the results are published on SQUAD leaderboard.

ELMO, *BERT*, *OpenAI GPT, GPT-2* and their variants are trained on large corpus of text with very deep models. The are mainly two downsides of these models: 1) Time required to train. 2) Hardware required to train. Since algorithms can be parallelised easily, more hardware leads to lesser time.

The cost required to train these models using a sophisticated hardware set up is very expensive[11] for an individual or an academic lab to afford for such use cases. These models are made publicly available because of the reason that they are expensive to recompute and fine tuning it leads to better reuse of the models for downstream

[11] https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/

tasks. But if one wants to pre-train these models from scratch, the cost involved is expensive.

There are new variants of the above models released very frequently these days by different research teams which can be accessible and tested easily by using a repository by Huggingface[12] who currently host 7 types of transformer models namely: 1)BERT, 2)GPT, 3)GPT-2, 4)Transformer XL, 5)XLNet, 6)XLM, 7)RoBERTa which can be used easily towards any NLP downstream tasks without the need of pretraining the models ourselves.

Moving on from SQUAD

The simplicity of SQUAD dataset was criticized and the assumption that the answer was always contained in the paragraph gives a pseudo positive relevance for all the paragraphs which in real time QA setting might not hold true because finding the "relevant" paragraphs which contains the answer is also a challenge.

We mainly discuss about the dataset of SQUAD in this section because the *Reading Comprehension* task witnessed a lot of models and progress in question answering. Several changes were proposed because of the issues of SQUAD dataset, and the authors also released a new dataset SQUAD 2.0 (Rajpurkar, Jia, et al., 2018) with additional questions without an answer.

Because of the different other assumptions and changes proposed to the task, Section 3.2.3 and 3.2.4 discuss either a different QA setting or a modified version of *Reading Comprehension* task.

3.2.3 OpenQA: back to the original QA task

OpenQA or *open domain question answering* is a QA task whose goal is to retrieve answer to a given question in open domain.

The input is only a question and a text collection, and the underlying systems should perform all possible operations to return a short answer. Such operations can be from the structured pipeline as explained in Figure 3.1. Traditional *Reading Comprehension* models are provided with a paragraph which contains the answer. Whereas in *OpenQA* there is no supporting paragraph provided to extract answer directly.

[12] https://github.com/huggingface/pytorch-transformers

A typical system would use an IR engine to retrieve relevant paragraphs for a question and perform *Answer Sentence Selection* to find the best set of sentences which contain the answer.

An answer extraction module such as a *Reading Comprehension* model can be used to extract an answer from the sentences and choose the best one among them but the assumption that the paragraph is always relevant (A relevant paragraph is a paragraph that is likely to contain the answer), does not apply for *OpenQA*. There can be relevant paragraphs which do not contain an answer, which makes *OpenQA* a more realistic and tougher problem than the task such as the one aimed by the *SQUAD* dataset.

Datasets

There are not many QA datasets for *OpenQA* which are large scale in size to use it effectively with a neural network model. For this reason works such as (K. Lee et al., 2019; Chen, Fisch, et al., 2017) convert a *Reading Comprehension* datasets into *OpenQA* ones just by considering their question and answer strings and ignoring the provided gold standard paragraphs and follow the IR method either in a cascaded manner (Chen, Fisch, et al., 2017) or by learning both the document retrieval and answer extraction model in an end to end fashion (K. Lee et al., 2019).

Dataset statistics and details can be found in Section 2.3 in detail.

Models and Approaches

A straightforward way to approach this task is by doing it in a strongly supervised manner where the model assumes that the paragraphs retrieved from a retriever model contain the answer and the noise is ignored. The authors of (Chen, Fisch, et al., 2017) do it in this way where the retriever model is used to retrieve relevant paragraphs for questions, which are further used in the reader model which is a *Reading Comprehension* model.

But in reality, there is always noise induced in the retrieved paragraphs as some paragraphs might not contain answers when an IR approach is used. As pointed out by (Singh, 2012), QA is fundamentally different from IR. (K. Lee et al., 2019) highlight that IR is concerned with lexical and semantic matching, but in QA the models require more language understanding, since users are explicitly looking for unknown information which may not be lexically or semantically closed to question terms. Works by (Joshi et al., 2017; Dunn et al., 2017; Dhingra, Mazaitis, and William W Cohen, 2017a) consider this noise and follow a weakly supervised manner which removes the strong supervision and considers the noise in the gold standard data by IR systems.

Models built for *OpenQA* ideally should consider this noise from the retriever. Therefore works by (Choi, Hewlett, et al., 2017; S. Wang, M. Yu, Guo, et al., 2018) attempt to consider this noise in the model of (Chen, Fisch, et al., 2017) by separating the question answering into paragraph selection and answer extraction. Both these models only select the most relevant paragraphs among all retrieved paragraphs to extract answers. By doing so, only the best scored paragraph from paragraph selection is used for answer extraction. This approach neglects information present in other paragraphs and does not take negative paragraph information into consideration by doing as above.

A follow up work by (S. Wang, M. Yu, Jiang, et al., 2017) propose strength-base and coverage-based re-ranking approaches for retrieved paragraphs, which can aggregate[13] the results extracted from each paragraph by an existing *Reading Comprehension* system like *DRQA* by (Chen, Fisch, et al., 2017) to better determine the answer. However, this still suffers from the noise issue in distant supervision data because it considers all retrieved paragraphs indiscriminately. The authors of (Y. Lin et al., 2018) use a paragraph selector to filter out noisy paragraphs and keep the best ones to perform answer extraction by considering the combined paragraph and answer probabilities.

The most popular pipeline QA system which raised media attention was the IBM Watson which won the jeopardy challenge against human competitors[14]. IBM Watson follows a parallel component based pipeline approach (Ferrucci et al., 2010) whose complex workflow is as shown in the Figure 3.7.

All the above discussed approaches rely on a cascaded style model which first performs paragraph retrieval and then followed by answer extraction. Both the processes are done sequentially and not learnt together. The authors of (Chen, Fisch, et al., 2017) use the IR engine like a blackbox model without learning any parameters for the model. The first end-to-end model which learns to retrieve evidence from an open corpus and supervise only by question answer string pairs is proposed by (K. Lee et al., 2019). The authors highlight that the main challenge

[13] Consider values calculated from different paragraphs for a question

[14] https://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html

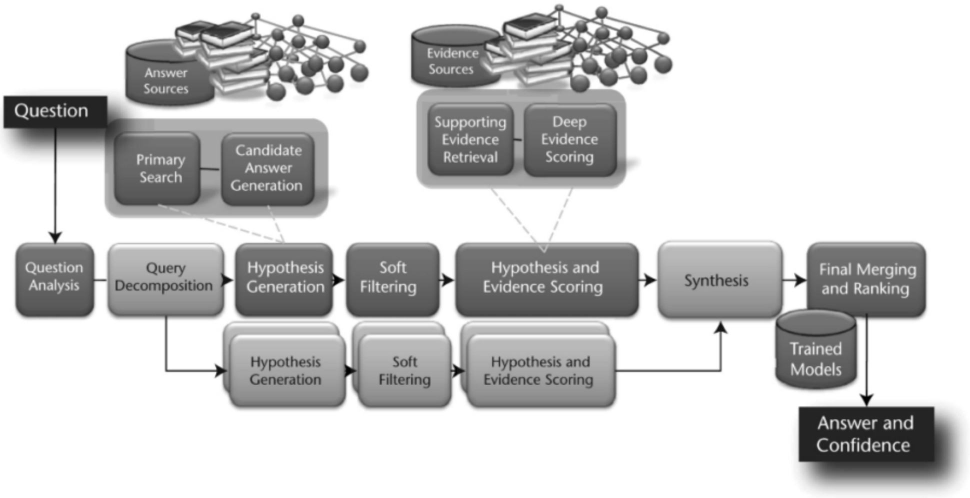
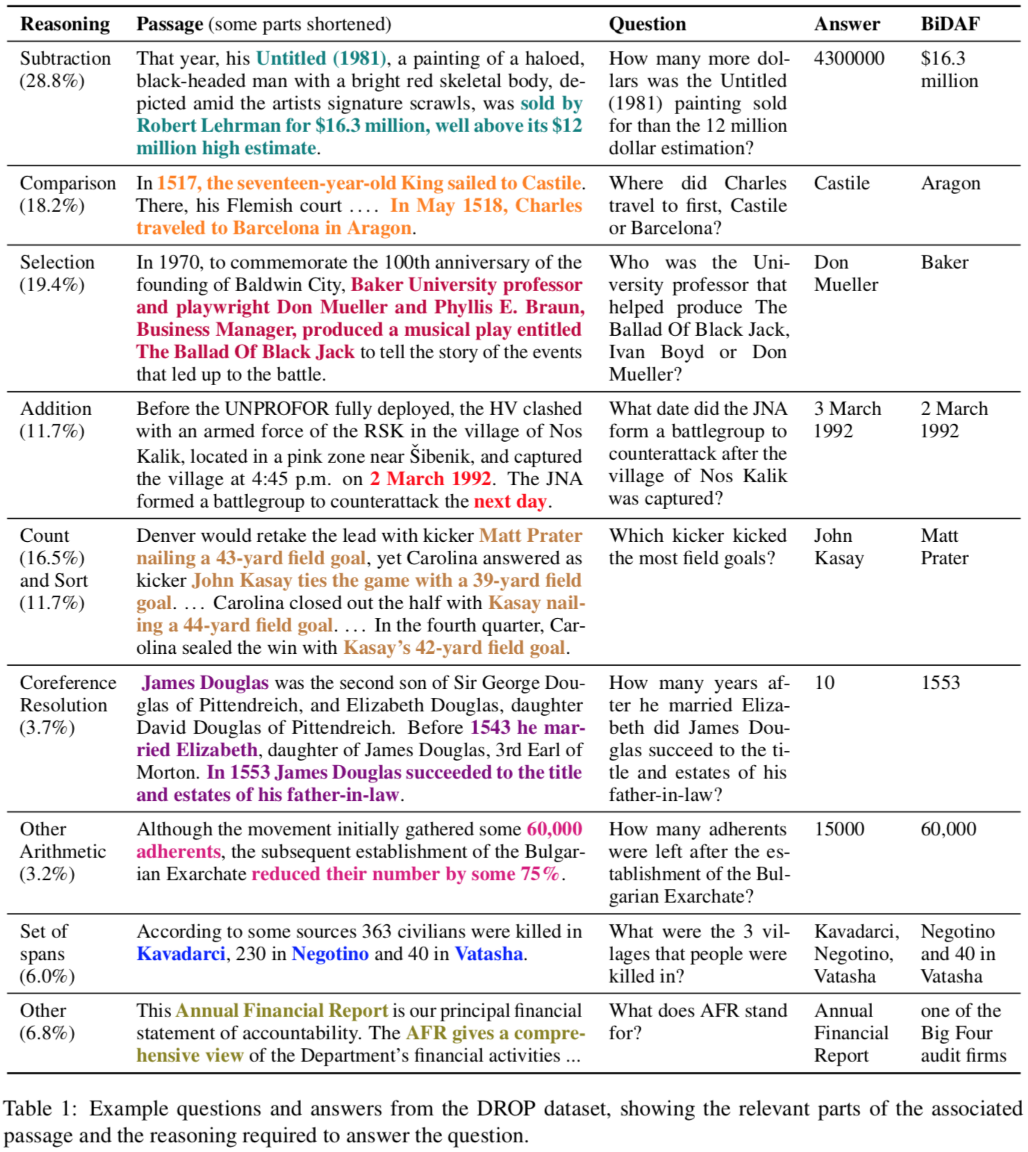


Fig. 3.7: Overall DeepQA architecture of the IBM Watson QA system

to build a fully end to end model is by considering the retrieval as a latent variable which is impractical to learn from scratch and using IR systems result in a potentially suboptimal starting point. So they propose a pre-training task called as Inverse Cloze Task (ICT) which is done in an unsupervised manner. Both the retrieval model which is pretrained with ICT and answer extraction model like BERT on *Reading Comprehension* are trained together end to end.

3.2.4 Reading Comprehension 2.0 - Modifications and Future of QA

Reading Comprehension task was mainly experimented on SQUAD v1.0 dataset as it was the first ever human annotated large scale dataset for the task. In section 2.2.4, some problems were discussed and new methods were also introduced which gave rise to new datasets. These new methods and datasets are explained in this section and we term it as *Reading Comprehension 2.0* as they are generally a variant of the original *RC* task.



Reasoning	**Passage** (some parts shortened)	Question	Answer	BiDAF
Subtraction (28.8%)	That year, his **Untitled (1981)**, a painting of a haloed, black-headed man with a bright red skeletal body, depicted amid the artists signature scrawls, was **sold by Robert Lehrman for $16.3 million, well above its $12 million high estimate**.	How many more dollars was the Untitled (1981) painting sold for than the 12 million dollar estimation?	4300000	$16.3 million
Comparison (18.2%)	In **1517, the seventeen-year-old King sailed to Castile**. There, his Flemish court .... **In May 1518, Charles traveled to Barcelona in Aragon**.	Where did Charles travel to first, Castile or Barcelona?	Castile	Aragon
Selection (19.4%)	In 1970, to commemorate the 100th anniversary of the founding of Baldwin City, **Baker University professor and playwright Don Mueller and Phyllis E. Braun, Business Manager, produced a musical play entitled The Ballad Of Black Jack** to tell the story of the events that led up to the battle.	Who was the University professor that helped produce The Ballad Of Black Jack, Ivan Boyd or Don Mueller?	Don Mueller	Baker
Addition (11.7%)	Before the UNPROFOR fully deployed, the HV clashed with an armed force of the RSK in the village of Nos Kalik, located in a pink zone near Šibenik, and captured the village at 4:45 p.m. on **2 March 1992**. The JNA formed a battlegroup to counterattack the **next day**.	What date did the JNA form a battlegroup to counterattack after the village of Nos Kalik was captured?	3 March 1992	2 March 1992
Count (16.5%) and Sort (11.7%)	Denver would retake the lead with kicker **Matt Prater nailing a 43-yard field goal**, yet Carolina answered as kicker **John Kasay ties the game with a 39-yard field goal**. ... Carolina closed out the half with **Kasay nailing a 44-yard field goal**. ... In the fourth quarter, Carolina sealed the win with **Kasay's 42-yard field goal**.	Which kicker kicked the most field goals?	John Kasay	Matt Prater
Coreference Resolution (3.7%)	**James Douglas** was the second son of Sir George Douglas of Pittendreich, and Elizabeth Douglas, daughter David Douglas of Pittendreich. Before **1543 he married Elizabeth**, daughter of James Douglas, 3rd Earl of Morton. **In 1553 James Douglas succeeded to the title and estates of his father-in-law**.	How many years after he married Elizabeth did James Douglas succeed to the title and estates of his father-in-law?	10	1553
Other Arithmetic (3.2%)	Although the movement initially gathered some **60,000 adherents**, the subsequent establishment of the Bulgarian Exarchate **reduced their number by some 75%**.	How many adherents were left after the establishment of the Bulgarian Exarchate?	15000	60,000
Set of spans (6.0%)	According to some sources 363 civilians were killed in **Kavadarci**, 230 in **Negotino** and 40 in **Vatasha**.	What were the 3 villages that people were killed in?	Kavadarci, Negotino, Vatasha	Negotino and 40 in Vatasha
Other (6.8%)	This **Annual Financial Report** is our principal financial statement of accountability. The **AFR gives a comprehensive view** of the Department's financial activities ...	What does AFR stand for?	Annual Financial Report	one of the Big Four audit firms

Table 1: Example questions and answers from the DROP dataset, showing the relevant parts of the associated passage and the reasoning required to answer the question.

Fig. 3.8: Table presented by (Dua et al., 2019) with different types of reasoning required for the QA in DROP dataset

Unanswerable questions

The goal of introducing unanswerable questions in SQUAD 2.0 by (Rajpurkar, Jia, et al., 2018) is to learn models which can determine if an answer exists to a question in the paragraph. This is similar to the questions of TRECQA task which did not contain any answers in the documents, but on a large scale.

In the setting of open domain question answering (OpenQA), an answer is not always present in the paragraph. There are relevant paragraphs in accordance with a question, which do not contain an answer.

A simple way to learn this datapoint as done by the authors of SQUAD v2.0 is by using the models of (Levy et al., 2017; Clark and Gardner, 2018) which learn to predict the probability that a question is unanswerable, in addition to extracting answers if the question is answerable. A binary classifier which predicts this probability can be learnt along with answer span extraction probabilities *Start* and *End* of the answer. Several works including one of the state of the art models BERT by (Devlin et al., 2018) uses the above method to handle this case.

Discrete reasoning over paragraphs

As the *RC* task is relatively simple and does not need any reasoning required to find answers in the paragraph, the authors of (Dua et al., 2019) introduce a new dataset named *DROP - Discrete Reasoning Over Paragraphs* which is a more challenging reading comprehension dataset which requires discrete reasoning to answer questions which demand numerical answers, choice answers etc. and not just lexically similar term matching answers.

Figure 3.8 shows different types of questions present in the DROP dataset which require different reasoning skills to answer the question. The baseline approach proposed by the authors uses a multi-class classification approach where the traditional *RC* type answer offsets are extracted along with two classes to predict if it is a *Count* type or *Arithmetic* type reasoning, followed by predicting the numbers in the text. Although this is a pretty naive way of doing limited set of operations, the authors state that this is a promising approach to combine neural network methods and symbolic reasoning. The authors also created a leaderboard[15] where one can upload their systems and the website evaluates on the hidden test set automatically online.

Multihop reasoning

The DROP dataset provides a dataset and explains a baseline approach of using multi-class classification approach to solve the reasoning problem. But the answers are still relying on a single supporting paragraph. i.e. Questions in DROP dataset do not use any information from other paragraphs to answer questions.

Suppose we have a question who needs data from different paragraghs to answer a question, neither DROP or other *RC* datasets have such data or methods to tackle this situation. To address this issue which is called as *Multihop reasoning*, (Zhilin

[15] https://leaderboard.allenai.org/drop/submissions/public

Yang, Qi, et al., 2018) introduce this problem and release a dataset called **HOTPOT QA** which addresses two main issues: 1) Multihop reasoning, 2) Sentence level supporting facts to provide explainability of the answers.

The systems should not only return answers to the questions which are inferred based on different supporting paragraph information, but also provide supporting facts which resulted in that answer prediction. The dataset also contains questions where two entities are compared, two entities are used to bridge another entity. The authors also propose a baseline approach which uses strong supervision over facts in a multi-task setting. The authors created a leaderboard[16] where one can upload their systems and the website evaluates on the hidden test set automatically online.

3.3 Evaluations and Challenges

A question answering system is tested always on datasets to determine its capability. Recently, there have been several leaderboards which compare results on a hidden dataset and report the performance of a system. These are online exclusive, some are automatic and most of them are updated frequently all over the year.

Some of such popular online leaderboards are:

- SQUAD v1.0 and v2.0 datasets by (Rajpurkar, J. Zhang, et al., 2016; Rajpurkar, Jia, et al., 2018)
- HotpotQA dataset by (Zhilin Yang, Qi, et al., 2018)
- DROP dataset by (Dua et al., 2019)
- Natural Questions dataset by (Kwiatkowski et al., 2019)

Offline Evaluation campaigns or challenges are held in order to gather research community efforts to solve a task by providing a standard dataset and evaluation metrics. Participating teams must submit either their systems (code + model) or the system predictions on a test set which the organizers will evaluate and compare with others. Usually a workshop is conducted at the end of the shared task where participants discuss their systems, approaches, tricks etc. and explain how their systems perform. To make it comparable, all the systems use the same datasets. In this section we discuss six such workshops or conferences based on question answering and mainly focus on explaining about the workshop, data used and

[16] https://hotpotqa.github.io

different tasks involved. However, this section does not provide an exhaustive list of all popular workshops involving question answering.

TREC

The TREC question answering (QA) track was the pioneer in large-scale evaluation of open-domain question answering systems. TREC is a workshop series designed to provide the infrastructure required for large-scale evaluation of text retrieval and related technologies. A "QA" track was introduced in TREC-8 in 1999 which went on for several years. The motivation was to foster research towards information retrieval systems than document retrieval systems.

Participants were given a document collection and a test set of questions, and expected to return a ranked list of five answer strings which contain an answer to the question. Human assessors read string and decided if it contained the answer or not. TREC-9 dataset contained around 500 questions which is a small scale dataset for today's standards.

In Trec QA task, QA systems return an actual answer (can be an answer sentence or a phrase), rather than a ranked list of documents, in response to a question. TREC has had a question answering track since 1999; in each track the task was defined such that the systems were supposed to retrieve small snippets of text that contained an answer for open-domain, factoid questions.

Trec QA evaluations considered the possibility that different people might have different ideas of what constitutes a correct answer. And the judgements by three individuals which were in conflict were decided by the official evaluation but the difference in the answer judgements were used to measure the judgement on system scores. More detailed version of the evaluation can be found in the document by (Voorhees, 2002).

These datasets from the past (1999-2004) are curated (questions without an answer were removed) by (M. Wang, Smith, et al., 2007) and used popularly as *TrecQA data for Answer Sentence Selection* today.

CLEF

The CLEF Initiative (Conference and Labs of the Evaluation Forum, formerly known as Cross-Language Evaluation Forum) is an ongoing initiative started in 1999 in the view of supporting and promoting research, innovation, and development of information access systems with an emphasis on multilingual and multimodal information with various levels of structure.

CLEF initiative is split into two parts: 1) Peer-reviewed conference 2) A series of evaluation labs with some datasets released to conduct evaluations. CLEF has different tracks such as ImageCLEF, VideoCLEF, QA@CLEF, CLEF-eHealth related to different data domains and tasks.

QA track at CLEF has different language question answering tasks. Questions in this track are in natural language. However, answering some questions may need to query Linked Data and some questions may need textual inferences and querying free text data. QA@CLEF had various types of QA tasks since 2003-2014. Some of the QA tasks in this track are, 1) BIOASQ - a biomedical challenge dataset which we use in our work. 2) QA4MRE - a QA task which has multiple choice questions related to a document. 3) QALD - a QA task on linked data. 4) Answer Validation Exercise to assess QA responses and decide whether an Answer to a Question is correct or not according to a given Text. More details can be found on the track site[17].

QA4MRE - Question Answering for Machine Reading Evaluation was a campaign which was held during 2011-2013 aimed at evaluating Machine Reading systems through Question Answering and Reading Comprehension Tests. The definition of *Reading Comprehension* is slightly different these days (also in our work) as the field of question answering refer to *Reading Comprehension* and *Machine Reading* as an answer extraction task, whereas *QA4MRE* was about detecting the right answer in multiple choices provided. Therefore the evaluation and the models built for both the tasks are different even though they carry the same name. More about QA@CLEF can be found on their website[18].

BIOASQ

BIOASQ[19] challenge is a large-scale biomedical semantic indexing and question answering task (Tsatsaronis et al., 2015) which has been successful for 7 years. The challenge proposes several tasks using Biomedical data. One of the tasks focuses on Biomedical question answering (Task B Phase B) where the goal is to extract answers for a given question from relevant snippets.

[17] http://nlp.uned.es/clef-qa/repository/
[18] http://www.clef-initiative.eu/track/qaclef
[19] http://bioasq.org/

Below is an example of a factoid question from the BIOASQ dataset for QA (task B). Each sample contains a question, answer and relevant snippets.

Question: What is the mode of inheritance of Wilson's disease?
Answer: autosomal recessive
Snippets: The overall sex ratio of patients was nearly 1:1,
and genetic analysis of 20 families confirmed an autosomal
recessive mode of inheritance.

Every year the challenge is organized in 5 batches, a training set with all the gold standard data is released earlier and the 5 test sets with all new questions and snippets are provided in each batch release. System predictions are expected to be submitted within 24 hours of release of the test set. Biomedical questions with their exact answers, relevant text snippets, concepts, articles, summaries were constructed or selected by biomedical experts from around Europe.

BIOASQ 7 is the seventh challenge and the evaluation measures for BIOASQ task B has always been the same. Strict Accuracy, Lenient Accuracy and Mean Reciprocal Rank (MRR) are the three evaluation measures used. To compute the scores, the exact match of strings between the predictions and the gold standard answers is used to decide if a system answer is correct. Strict accuracy is the rate of top 1 exact answers. Lenient accuracy is the rate of exact answers in top 5 predictions. MRR is the mean reciprocal rank computed on the top 5 system answers. These measures have been the same since the 1st challenge, although the first four challenges had triples and concepts along with snippets in the data. In the last two challenges, only relevant snippets for questions are released.

Several works in the past BIOASQ tasks have used classical question answering pipeline architecture adapted to the biomedical domain which includes modules such as question analysis, passage selection, answer selection which contribute towards the extraction of suitable answers. Some use the domain-specific information from UMLS tools such as Metamap (Schulze et al., 2016), along with other NLP tools like Corenlp, LingPipe (Zi Yang et al., 2016).

One of the first attempts to use deep learning algorithms for the BIOASQ task was reported in BIOASQ 5 by (Wiese et al., 2017c) where the dataset was adapted to be used as a reading comprehension dataset whose goal is to extract answers from snippets. The authors use a model trained on open domain questions, and perform domain adaptation from open domain to biomedical domain using BIOASQ data. The models prior to the first deep learning model by (Wiese et al., 2017c) got lower scores than the deep learning ones.

Starting from BIOASQ 7 in 2019, BERT models are being used with different training strategies such as training with varying paragraph data and varying data the BERT model is pre-trained on for language modelling task. BIOBERT used Biomedical data to pre-train BERT model and use it for BIOASQ task (J. Lee et al., 2019). And authors of (Hosein et al., 2019) use BERT model on BIOASQ task by pre-training BERT on Natural Questions dataset. Both BIOBERT and work by (Hosein et al., 2019) obtain similar results on same sets and BIOBERT being generally better than others. Both the models first train on open domain reading comprehension task and perform domain adaptation in their best scored approaches.

MediQA

The MEDIQA challenge[20] aims to attract further research efforts in Natural Language Inference (NLI), Recognizing Question Entailment (RQE), and their applications in medical Question Answering (QA).

The shared task of the workshop had a QA task which is to filter and improve the ranking of automatically retrieved answers. Similar to *Answer Sentence Selection*, the MEDIQA task had documents (collection of paragraphs) instead of a sentence or a small paragraph as an answer. For a given question a system had to return if a question and an answer document pair is correct or wrong (0 or 1). The dataset contained 104 consumer health questions and 104 simple questions about diseases. MediQA had an automatic evaluation system[21] setup by the organizers where the participants could upload the test set predictions.

An overview of the workshop is presented by (Ben Abacha et al., 2019) where more details about other tasks can also be found.

MRQA

Machine Reading for Question Answering (MRQA) workshop[22] was organized for the first time in 2018 to gather researchers to address and discuss important research topics surrounding Machine Reading (Reading Comprehension). The goal was to discuss aspects such as Accuracy, Interpretability, Speed, Scalability, Robustness, Dataset Creation, Dataset Analysis, Error Analysis. The first task mainly covered

[20] https://sites.google.com/view/mediqa2019

[21] https://www.aicrowd.com/challenges/mediqa-2019-question-answering-qa

[22] https://mrqa2018.github.io/

works such as new models, datasets, training methods, error analysis etc. on *RC* task.

The second workshop along with being open to normal papers also had a shared task where the goal was to evaluate the generalisation of models beyond the datasets they are trained on. Intuition is that the models should do more than merely interpolate from the training set to answer test examples drawn from the same distribution but rather extrapolate on data from different distributions. A training set was created by pooling six large scale datasets and test sets were ten different test sets to test the generalisation. More details can found on the link[23].

An overview and proceedings of the workshop is presented by (Choi, Seo, et al., 2018) which also lists different works published recently.

3.4 Conclusion

In this chapter, we have detailed about question answering models and approaches in the past, and currently used state-of-the-art systems. Indeed, the definition of question answering has evolved in many different ways, resulting in various datasets with different challenges and a large number of models with their distinct advantages and limits. According to the issues we address, we experiment on different QA tasks, including *Reading Comprehension*, *Answer Sentence Selection*, and *OpenQA*.

The current state-of-the-art models on question answering tasks mainly use neural network based approaches. They are mainly proposed for open domain with large datasets. In contrast, the interests in neural network models for domain specific datasets, that are small scaled datasets, are much less studied. One way of using neural network models for small sized datasets is via domain adaptation. We will explore different possibilities for doing domain adaptation and experiment different neural network models and different pre-training data for BIOASQ dataset.

The *Reading Comprehension* task mainly used in the context of SQUAD dataset (Rajpurkar, J. Zhang, et al., 2016) assumes that the paragraphs given along with the questions always contain an answer. This assumption does not hold good in the case of the BIOASQ dataset (Tsatsaronis et al., 2015) where there are several paragraphs that do not contain an answer. Several works on BIOASQ such as (Wiese et al., 2017b; Yoon et al., 2019; Hosein et al., 2019) use *Reading Comprehension* task modelling for BIOASQ where the paragraphs that do not contain an answer are ignored from the data. The pre-training is done on SQUAD v1.0 dataset and

[23] https://mrqa.github.io/shared

fine-tuning on the BIOASQ data in all the previous works. We believe that modelling this task differently considering all kinds of paragraphs from BIOASQ dataset would improve the performance of the models.

BERT model (Devlin et al., 2018) is used in many QA tasks currently. Previous best scoring models, such as (Hosein et al., 2019; Yoon et al., 2019), fine-tune the BERT language model (primary task of BERT) towards the target QA tasks like BIOASQ. (Yoon et al., 2019) pre-train on SQUAD v1.0 dataset (Rajpurkar, J. Zhang, et al., 2016) and (Hosein et al., 2019) pre-train on Natural Questions dataset (Kwiatkowski et al., 2019) before fine-tuning on BIOASQ dataset. Our intuition is that the choice of the dataset to pre-train plays a major role on the performance on BIOASQ dataset. We will study the variation of performance when a QA model is pre-trained on different datasets.

The performance of neural network models is based on the dataset annotation and evaluation protocols. BIOASQ dataset lacks many variants of answer in the gold standard dataset annotated by human experts. Since it is important to be able to construct annotated corpus of a high quality at a reasonable cost, we will study how structured resources, like UMLS which contain large amount of information on biomedical domain, can be used to annotate the answer variants in order to improve the performance of the QA models.

Since end-to-end models are one of the goals of using neural networks, not much efforts have been put towards improving the outputs of a neural network model by post-processing the predictions. Our hypothesis is that the semantic features and structured information from different paragraphs can be used to post-process the predictions to further increase the performance. In the biomedical domain, there exists structured information such as UMLS which can help in detecting biomedical entities and find the matching types between question and paragraphs. In the open domain, the *Expected Answer Types* inferred from questions can help in finding the matching entity types between question and paragraphs. These matching types highlight the entity types and the entities which are likely to be the answers for the questions. We intend to study and use these features in different ways to improve the QA performance on different models and tasks.

In the following chapters, we detail about our two research questions of 1) Building better models for large scale and small scale data and 2) Leveraging structured and semantic information into QA models and present several hypothesis, experiments, and result comparison in detail.

4 Building Models for Small Scale and Large Scale Datasets

"Deep neural network models perform better with more data!", "One needs more data to use a deep neural network model", "Less data? Not suitable to use deep learning models!, "Why is it difficulty to get annotated datasets?" are some of the common phrases we come across in recent times.

In our work, one of the goals was to build a better question answering model for domain specific data. *Using deep neural network models directly on small scale datasets would not fetch the best scores because of the small scale nature of the datasets* - is one of the popular beliefs among the deep learning community. Since the data (BIOASQ dataset) we came across was small scale, we wanted to experimentally determine how bad the performance of a deep neural network model (usually targeted for large scale datasets) would be on this small scale dataset.

One of the solutions to obtain optimal performance on small scale datasets is to train the models on large scale datasets, and then fine-tune (retrain the model) it towards the small scale datasets. This allows reusing the state-of-the-art models for small scale QA datasets. This process is called as *Domain Adaptation* and we present a method to perform domain adaptation from open domain data (which is generally large scale) towards biomedical domain data (which is generally small scale).

In this chapter we report several experiments on small scale domain specific dataset of BIOASQ task. We define the process of domain adaptation by formalizing the terms such as *Pre-training* (first training) and *Fine-tuning* (retraining), and explain three main kinds of domain adaptations.

Choosing a good model among several state-of-the-art models is a challenging task. We present certain factors which are important to consider before choosing to work on their implementations and explain why we choose one model over the others for *Reading Comprehension task*.

To perform domain adaptation, one must first choose a suitable QA task to perform the first training phase with a large scale dataset. We compare two QA task models (One for *Reading Comprehension task* and one for *Open domain question answering* task) for training and show one outperforms the other on BIOASQ dataset. We also hypothesize that using different datasets (sometimes more than one combined together) to pre-train a model impacts the downstream finetuning performance. To show this behaviour we pre-train a single model with different datasets and finetune it to biomedical data.

The organisation of this chapter is as follows:

- We first define and detail the state-of-the-art question answering models used in our work for 1) Reading comprehension, 2) Open domain question answering models, 3) Answer sentence selection tasks.

- We detail the concept of domain adaptation, and show how to adapt these models to biomedical domain along with general notions of pre-training and fine-tuning.

- We experiment with different word embedding spaces for a QA model to determine which performs better.

- We explain about the choice of a good QA model and compare two QA task models for pre-training.

- For performing domain adaptation, we also need a large scale dataset to pre-train the models, we compare with several datasets and report performance on domain adaptation.

4.1 Question Answering models

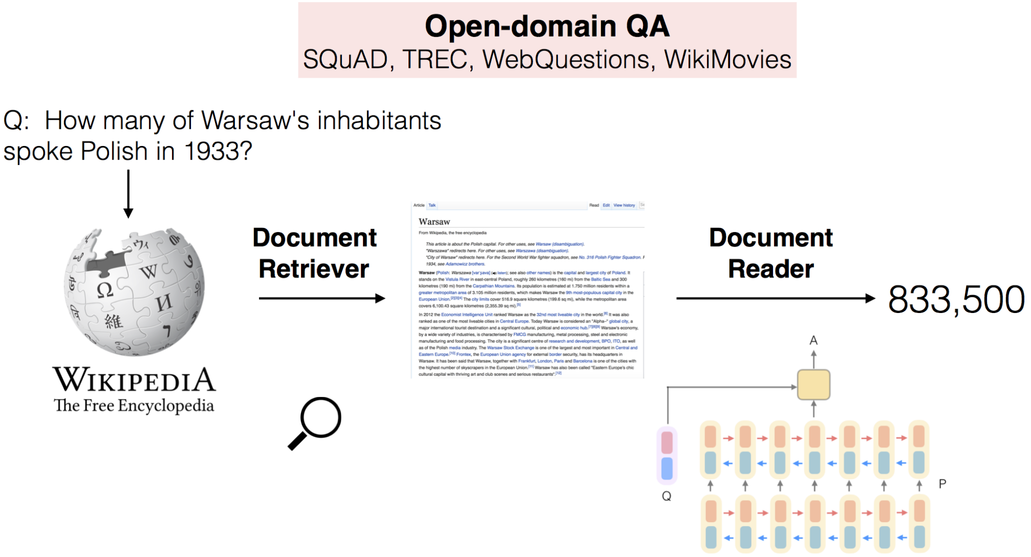
Question Answering is a broad domain of natural language processing which can be defined in different ways based on the types of tasks involved. As introduced in chapter 2, there are several tasks under question answering.

In this section we explain two QA models which we have used extensively in several studies throughout this research work.

- Reading comprehension, a task that deals with extracting an answer from a paragraph. The input data would ideally contain a question and a paragraph. The model is expected to predict an answer span in the paragraph.

- Open Question Answering, a task that deals with finding an answer provided a question. Sometimes there is a collection of texts provided from an IR engine and sometimes there is no paragraph provided along with the question.

4.1.1 Reading Comprehension



Open-domain QA
SQuAD, TREC, WebQuestions, WikiMovies
Q: How many of Warsaw's inhabitants spoke Polish in 1933?
WIKIPEDIA
The Free Encyclopedia
Document Retriever
Warsaw
Document Reader
833,500
A
Q
P

Fig. 4.1: *DRQA* model by (Chen, Fisch, et al., 2017)

Reading Comprehension or *RC* task although existed since several years, it has gained a lot of attention since the release of SQUAD dataset (Rajpurkar, J. Zhang, et al., 2016). Many works on *Reading Comprehension* or *RC* task can be found on the SQUAD leaderboard[1].

We chose a model named *DRQA* by (Chen, Fisch, et al., 2017) for the following three reasons. 1) The document reader model of *DRQA* that performed the best on SQUAD dataset at the time of experimenting, 2) The code was released publicly[2] to replicate the results reported in the paper, 3) The model was training rapidly (4 hours approximately on a single GPU) and performed better than models which took longer to train and were more complex.

DRQA has two models namely *Retriever* and *Reader* which work in a cascaded fashion when the questions are not provided with related paragraphs. Figure 4.1 shows the workflow of the overall model.

[1] https://rajpurkar.github.io/SQuAD-explorer/
[2] https://github.com/facebookresearch/DrQA

The retriever model is primarily based on a tool which uses information retrieval (non-machine learning) techniques to retrieve relevant paragraphs (or documents) for the question based on question terms. The reader model is a simple Bi-LSTM model for *Reading Comprehension* task which takes as input a question and a paragraph and aims at extracting an answer from the paragraph.

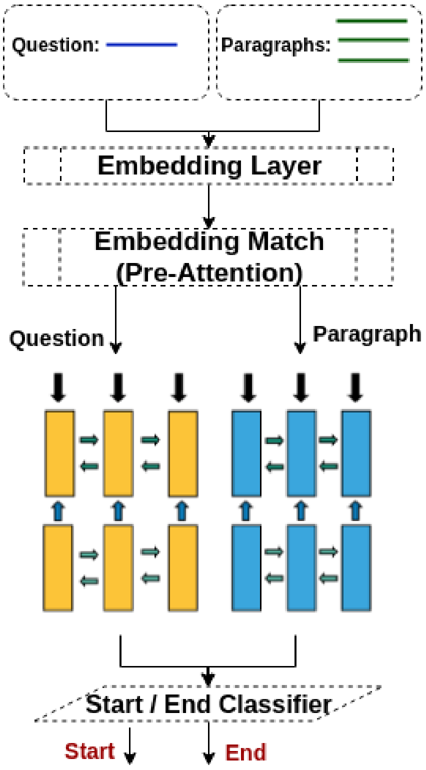


Fig. 4.2: Detailed reader model of DRQA

For our work, we only use the reader model of *DRQA* as we have supporting paragraphs for questions for this task. An overview of the reader model is presented in the figure 4.2.

The input to the reader model are sequences of question words and paragraph words. Both the question and the paragraph strings are tokenized and their word embeddings are used by the model i.e. Question words $Q = \{q_1, ..., q_m\}$ and paragraph words $S = \{s_1, ..., s_n\}$ are sequences which are encoded using an embedding layer of dimension D. The authors use Glove vectors by (Pennington et al., 2014) in their work so we use the same for this model.

$$E(Q) = \{E(q_1), .., E(q_m)\} \tag{4.1}$$

$$E(S) = \{E(s_1), .., E(s_n)\} \tag{4.2}$$

A pre-attention mechanism captures the similarity between paragraph words and question words in the same layer. We call it as pre-attention mechanism because it is applied before the embeddings are encoded with LSTM layers, which is usually how attention mechanisms are used such as in (Seo et al., 2016). For this purpose, a feature $\mathcal{F}align$ shown in Equation 4.3 is added as an input feature to the LSTM layer.

$$\mathcal{F}align(p_i) = \Sigma_j a_{i,j} E(q_j) \tag{4.3}$$

Where $a_{i,j}$ is,

$$a_{i,j} = \frac{exp\,(\alpha(E(s_i)) \cdot \alpha(E(q_j))}{\Sigma_{j'}\ exp(\alpha(E(s_i)) \cdot \alpha(E(q_{j'}))} \tag{4.4}$$

which computes the dot product between nonlinear mappings of word embeddings of question and paragraph.

They are followed by a 3-layer Bidirectional LSTM layers for both question and sentence encodings. Maximum length of the sequences are set as 200 tokens.

$$\{E(q_1), .., E(q_n)\} = \text{Bi-LSTM}(\{\tilde{E}(q_1), .., \tilde{E}(q_n\}) \tag{4.5}$$

$$\{E(s_1), .., E(s_n)\} = \text{Bi-LSTM}(\{\tilde{E}(s_1), .., \tilde{E}(s_n\}) \tag{4.6}$$

These LSTM states are connected to two independent classifiers that use a bilinear term to capture the similarity between paragraph words and question words and compute the probabilities of each token being start or end of the answer span.

$$P_{\text{start}}(i) \propto \exp\left(\mathbf{p}_i \mathbf{W}_s \mathbf{q}\right) \tag{4.7}$$

$$P_{\text{end}}(i) \propto \exp\left(\mathbf{p}_i \mathbf{W}_e \mathbf{q}\right) \tag{4.8}$$

During prediction, the best spans are chosen from token i to token i' such that $i \leq i'$ $\leq i + 15$ and $P_{start}(i) \times P_{end}\ (i')$ is maximized.

The combined probability predicted as shown above results in a set of predictions and for official evaluations like SQUAD, we only consider Top-1 prediction. For tasks like BIOASQ, Top-5 can be chosen based on the decreasing order of combined probability scores.

4.1.2 Pre-trained Large scale Language Models - BERT

Since the introduction of self attention mechanism by (Vaswani et al., 2017) which are usually termed as *Transformer models*, a lot of attention has been given towards using these as building blocks in several neural architectures such as in (A. W. Yu et al., 2018) for question answering. Pre-trained language models which are trained on large scale architectures like BERT by (Devlin et al., 2018) use transformer models as building blocks instead of LSTMs such as in ELMO by (M. Peters et al., 2018b).

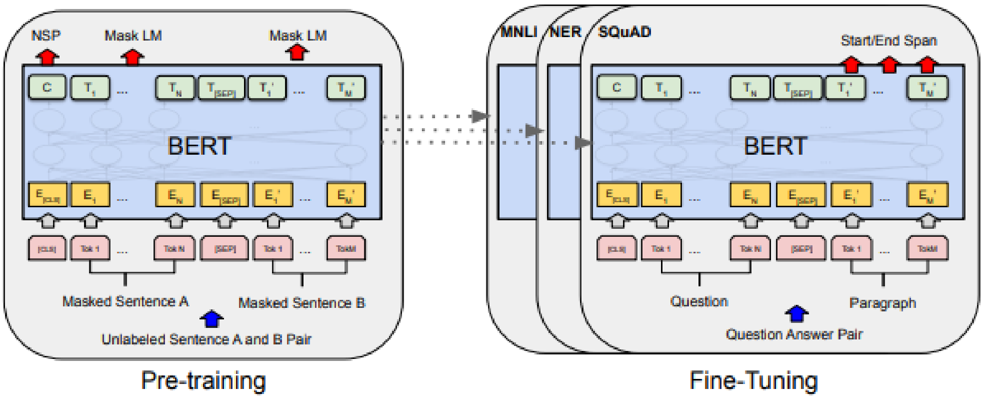


Fig. 4.3: BERT model modified for several NLP tasks by (Devlin et al., 2018)

The authors of BERT train two large scale neural language models (BERT Base and BERT Large) using transfer models as building blocks on two tasks namely 1) Masked Language Modelling and 2) Next Sentence Prediction. Further they modify the model to be able to work on several NLP tasks such as question answering, natural language inference, named entity recognition etc. They do this by modifying the final output layer to fit into target task. In our work we only use the fine-tuning modifications on a question answering task.

Figure 4.3 shows the final layer of BERT model modified into *Reading Comprehension* task SQUAD. Both the question and paragraph are packed as a single packed sequence with question and paragraph as two different embedding representations from BERT. Both *Start* and *End* which represents the answer span in the paragraph are two vectors $S \in \mathbb{R}^H$ and $E \in \mathbb{R}^H$ which are used only during fine-tuning for SQUAD task. The probability of word i being the start of the answer span is computed as a dot product between T_i and S followed by a softmax over all of the words in the paragraph:

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}} \tag{4.9}$$

The score of a candidate span from position i to position j is defined as $S \cdot T_i + E \cdot T_j$, and the maximum scoring span where $j \geq i$ is used as a prediction. By doing this small modification and fine-tuning the whole BERT model with this new layer at the end, the authors get state of the art results in several NLP tasks including *Reading Comprehension* on SQUAD dataset.

BERT has two models namely *BERT Large* and *BERT Base* which defines the number of layers and units used in the model. In our experiments, we only use *BERT Base* as it is faster and easier to fine-tune on a single GPU machine. *BERT Large* performs the best out of the two models but since our aim was not to fetch the best scores possible, we rely on *BERT Base* model for all of our experiments.

We use the open source code for BERT experiments from HuggingFace Inc.[3] by (Wolf et al., 2019), whose code is compatible with PyTorch and also is modifiable for several other NLP tasks models.

4.1.3 Open QA - Open Domain Question Answering Model

Open QA task is a straightforward question answering task whose input is just a question phrase without any additional information. One way of modeling this task which we use in our work is by using a *Retriever model* to retrieve relevant paragraphs for the question, followed by using a *Reader model* to extract answers from the retrieved paragraphs.

We present a model named *PSPR - Paragraph Selector and Paragraph Reader* which is an Open QA model by (Y. Lin et al., 2018) whose overview is presented in the Figure 4.4 and the code is available online[4]. We do not use the *DRQA* model as explained in section 4.1.1 directly for this task because *DRQA* works only on *Reading Comprehension* task which assumes that all questions are provided with relevant

[3] https://github.com/huggingface/transformers

[4] https://github.com/thunlp/OpenQA

paragraphs, which is not the case for *Open QA* task which contains a mixture of irrelevant and relevant paragraphs.

This model has two parts, namely Paragraph Selector (PS) and Paragraph Reader (PR) in a cascade fashion, that is why we named it as *PSPR* model. Normally this model is termed as *OpenQA* model. The authors of (Y. Lin et al., 2018) use *DRQA* model in this model for modelling the *Paragraph Reader (PR)*.

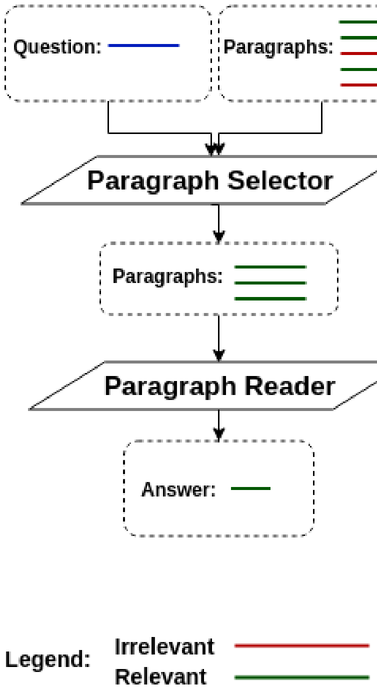


Fig. 4.4: OpenQA model by (Y. Lin et al., 2018)

Not all the paragraphs are relevant or contain the answer in the *OpenQA* task and also several paragraphs might contain the answer, therefore the paragraph probability computed by the *Paragraph Selector (PS)* model and the answer probability computed by the *Paragraph Reader (PR)* model are used to compute a combined probability.

Paragraphs for the questions are retrieved using an information retrieval model where some paragraphs are relevant and some are not. The Paragraph Selector model predicts a probability distribution $Pr\left(p_i|q,P\right)$ over all the retrieved paragraphs where P is the set of paragraphs for the question.

The Paragraph Reader model extracts answer spans as shown in the *DRQA* model for *Reading Comprehension* task and predicts a probability $Pr\left(a|q,p_i\right)$ for each answer span where p_i is i^{th} paragraph in Paragraph set P.

The reader model gives two probabilities (one for start and one for end token given by two classifiers) as described in equation 4.7 and 4.8. The answer probability $Pr(a|q, P)$ is computed as shown below:

$$Pr\left(a|q, p_i\right) = \sum_j Pr_s\left(a_s^j\right) Pr_e\left(a_e^j\right) \tag{4.10}$$

The Paragraph Selector uses tokenized question words $Q = \{q_1, ..., q_m\}$ and tokenized paragraph words $P = \{p_1, ..., p_n\}$ which are encoded using an embedding layer of dimension D.

$$E(Q) = \{E(q_1), .., E(q_m)\} \tag{4.11}$$

$$E(P) = \{E(p_1), .., E(p_n)\} \tag{4.12}$$

A RNN layer encodes the contextual information of the sequence.

$$\{E(q_1), .., E(q_m)\} = \mathrm{RNN}(\{\tilde{E}(q_1), .., \tilde{E}(q_m\}) \tag{4.13}$$

$$\{E(p_1), .., E(p_n)\} = \mathrm{RNN}(\{\tilde{E}(p_1), .., \tilde{E}(p_n\}) \tag{4.14}$$

Using this hidden representation, a self attention operation is applied to get the question representation q:

$$\hat{\mathbf{q}} = \sum_j \alpha^j \hat{\mathbf{q}}^j \tag{4.15}$$

where α^j encodes the importance of each question word against the other question words which is calculated as:

$$\alpha_i = \frac{\exp\left(\mathbf{w}_b \mathbf{q}_i\right)}{\sum_j \exp\left(\mathbf{w} b \mathbf{q}_j\right)} \tag{4.16}$$

Where w is the learnt weight vector. Finally, the probability of each paragraph is calculated via a max-pooling and a softmax layer as shown below:

$$Pr\left(p_i|q, P\right) = softmax\left(\max_j\left(\hat{\mathbf{p}}_i^j \mathbf{W} \mathbf{q}\right)\right) \tag{4.17}$$

where **W** is a learnt weight matrix.

The softmax operation in equation 4.17 is applied over total number of paragraphs per question therefore a probability value is predicted for each paragraph. Since not all the paragraphs contain an answer in the Open QA setting, the probability scores from equation 4.17 should indicate if there exists an answer or not.

While training, the paragraphs containing the answer are highlighted as 1 and the rest as 0. While testing, the best answer candidate ($\hat{a}$ from equation 4.19) is chosen with the highest probability $Pr(a|q, P)$ for a question q which is calculated as shown below :

$$Pr(a|q, P) = \sum_{p_i \in P} Pr\left(a|q, p_i\right) Pr\left(p_i|q, P\right) \tag{4.18}$$

$$\hat{a} = \arg\max_{a} \Pr(a|q, P) \tag{4.19}$$

For each paragraph, the authors of (Y. Lin et al., 2018) extract 10 answer spans based on their decreasing order of probabilities represented in equation 4.18. Choosing a lesser value instead of 10 might affect the model performance. This is an important hyperparameter to control the number of answer predictions per paragraph.

In the implementation provided online[5], the training process is done in three phases:

1. Pre-training the reader model to determine which paragraph has an answer and which doesn't based on answer presence.

2. Pre-training the selector model to learn a ranking function which is similar to the one in *Answer Sentence Selection task*.

3. Training an overall model using the above two pre-trained models to perform *Reading Comprehension* by combining paragraph and answer probabilities.

4.1.4 Choosing a *Good* model

What defines as a good model? Why is it hard to choose a good model? What factors should one consider before choosing a model for training? - these are some of the questions we asked ourselves before we began performing experiments on QA tasks.

[5] https://github.com/thunlp/OpenQA

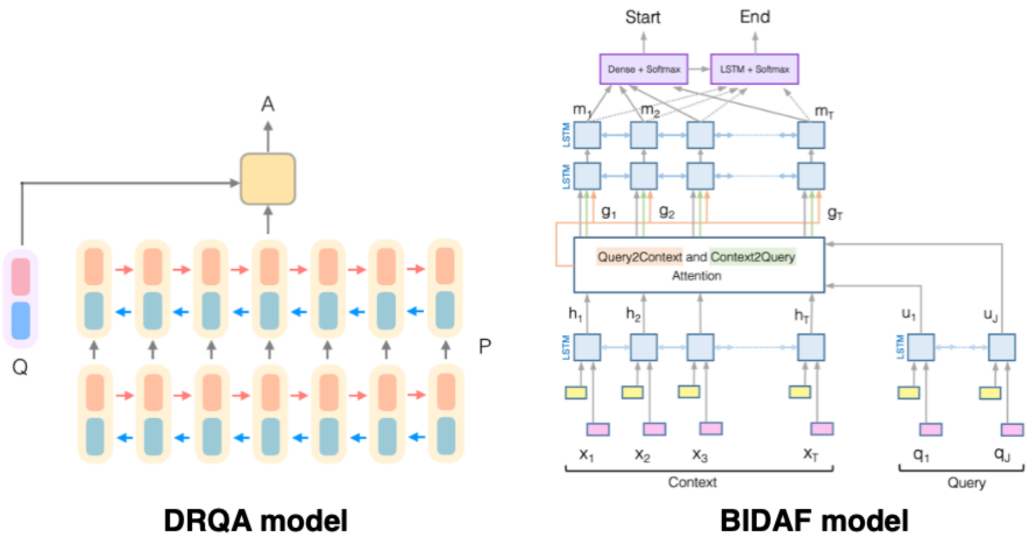
With the rise of deep learning models for many NLP tasks including the ones for question answering we have listed above, there have been exploding number of research articles and models for various tasks. It is often hard to catch up with the latest state-of-the-art research work. If someone wants to use a model to compare for the baseline performance, there has to be certain points to consider before jumping on using the latest or more visibly famous (on social media such as facebook and twitter) research models.

Here are some factors which are important to address:

1. Time required to train these models.

2. Computation power needed.

3. Model complexity and difference in performance over simple models.



DRQA model

BIDAF model

Fig. 4.5: DRQA model by (Chen, Fisch, et al., 2017) (Left) and BIDAF model by (Seo et al., 2016) (Right)

In an academic research lab scenario, we are often limited to smaller and basic computation resources required to run some experiments. This becomes the first most important aspect in choosing the right experimental setup and models.

Secondly, based on the complexity of the models, some models consume more time to train compared to some other simpler models who train faster. Therefore time is also an important aspect which contributes to choosing the right model needed for some experiments.

Model complexity and difference in performance over simple models is one of the aspects which seems to be less of a concern while experimenting with latest research works. When we began our work on *Reading Comprehension* task. We came across several works which were very similar to each other in terms of model complexity and their performance. For a QA task, research done on SQUAD dataset and the scores reported on the SQUAD leaderboard[6] are a good example to show this point.

Some models were better than the rest and simpler. Particularly we stumbled upon two models, A simple model *DRQA* by (Chen, Fisch, et al., 2017) which we have explained above and we use extensively in our works and another complex model named *BIDAF - Bidirectional Attention Flow* by (Seo et al., 2016).

BIDAF proposes a bidirectional attention flow and a complex RNN based architecture (atleast 2 LSTM layers) whose code[7] takes ∼20 hours to compute on the SQUAD train set and fetches **67.7%** EM score on the SQUAD dev set. *DRQA* which is a much simpler model with just 1 layer of 3 layered Bi-LSTM trains on SQUAD train set in ∼4 hours and fetches **69.5%** EM score on the SQUAD dev set. The training time of *DRQA* model is 5 times lesser than that of the *BIDAF* model.

These two models were reported and published around the same with *DRQA* being released in March 2017 and *BIDAF* being released in November 2016 with a gap of around 5 months. Although *DRQA* was published late, the model is much simpler and fetches better scores than *BIDAF* on the exact same dataset. This difference in model complexity and simpler model (*DRQA*) performing better on a comparable experimental setup on the same dataset raises suspicions about the actual contribution of the attention layers and extra LSTM layers required for the task as presented by (Seo et al., 2016).

Does this phenomenon occur because of engineering tricks which was done on the *DRQA* code or just hyper parameter optimization ? We could not come up with a conclusion in this regard so we went ahead with choosing *DRQA* as a base model for all our *Reading Comprehension* experiments. This phenomenon shows that using a simple model sometimes is better than using the latest and famous (well discussed in social media) models which may not be always better performing. Therefore we proceed by using a simple model for our experiments.

In the following section, we introduce *domain adaptation* and detail about adapting QA models which are primarily modelled for open domain question answering tasks, towards biomedical domain question answering.

[6] https://rajpurkar.github.io/SQuAD-explorer/

[7] https://github.com/allenai/bi-att-flow

4.2 Domain Adaptation

Domain adaptation is a field associated with machine learning and transfer learning. This scenario arises when we aim at learning from a source data distribution a well performing model on a different (but related) target data distribution. - Wikipedia[8]

In our research context, we use domain adaptation to learn a model on a large scale dataset and use the same model and its parameters to learn on a small scale dataset. In this case both the datasets are similar in nature but come from different source domains. This facilitates to use a deep neural network model effectively on small scale datasets like BIOASQ data. Training a model from scratch on a small scale dataset might not result in the best performance, therefore domain adaptation is carried out.

We first present the general process of domain adaptation before presenting different types of domain adaptations (sometimes referred as transfer learning) for biomedical QA task BIOASQ from open domain data.

4.2.1 The process

In this section we first briefly explain the process of domain adaptation in a generic manner. In the context of domain adaptation and transfer learning for deep neural networks, two terms are often used. 1) *Pre-training* - is a learning or training process of a model with randomly initialized model weights. 2) *Fine-tuning* - is a learning or training process but initialized from the model weights of the pre-trained model and not randomly initialized model weights. Both pre-training and fine-tuning together can be termed as *Domain Adaptation* when the domain of the data used for pre-training and fine-tuning are different. For example, open domain and biomedical domain in question answering.

Pre-training and fine-tuning or domain adaptation can also be done in several ways. The general approaches are listed below.

- *Type 1* - The target task remains the same for pre-training and fine-tuning. Pre-training should be done on a large scale dataset from random initialization of parameters. Fine-tuning can be done on a small scale dataset by loading the model parameters from pre-trained model rather than random initialization. This approach is used when a target dataset is small scaled and using it to train a deep neural network would result in overfitting. This type of pre-training

[8] https://en.wikipedia.org/wiki/Domain_adaptation

is common in computer vision field where models are pre-trained on Imagenet (Russakovsky et al., 2015) and fine-tuned on target image classification datasets.

- *Type 2* - The tasks are different for pre-training and fine-tuning. Pre-training should be done on a large scale dataset from random initialization of parameters. Fine-tuning should be done on a different model which uses certain parameters from the pre-trained model which are frozen (non-trainable) and learns some parameters which are randomly initialized on a different task. These approaches in NLP were initially proposed for sequence labelling tasks by (M. E. Peters et al., 2017) which were later evolved into ELMO (Embedding Language Models) by (M. Peters et al., 2018a) which significantly improved the state of the art across a broad range of challenging NLP tasks such as question answering, textual entailment and sentiment analysis. This type of method uses special contextual text embeddings obtained from the pre-trained models that are added as features into downstream models built for another task.

- *Type 3* - The tasks are different for pre-training and fine-tuning. Pre-training should be done on a large scale dataset from random initialization of parameters. Fine-tuning should be done on the pre-trained model by modifying certain layers to fit to the new task. Newly added layers can be randomly initialised and pre-trained model layers together with newly added ones are trained on the new task. This approach is similar to *Type 2* approach with a difference that the reference model can be slightly modified for target task rather than building a different model. This type of approach proposed by (Devlin et al., 2018) is being widely used in NLP tasks such as question answering, textual entailment, sentiment analysis, named entity recognition, relation extraction etc. which are easily done by modifying a final output layer of the original model and fine-tuned. Fine-tuning can be done either by learning the whole model parameters or learning only a part of the model by freezing the rest.

4.2.2 Biomedical Domain Adaptation for Reading Comprehension

As explained earlier, a neural network model like *DRQA* cannot be used in a straightforward way on a small scale dataset like BIOASQ dataset whose statistics are presented in table 4.1. The statistics compares BIOASQ data with two other large scale datasets and shows the difference in the number of samples. The large scale datasets *SQUAD v1.0* and *QUASAR-T* and several others which are currently used in different Question Answering tasks are at least 100 times larger than BIOASQ dataset. An assumption with deep neural network models is that it needs large scale datasets to perform better. We test this assumption by experimenting with a deep neural network model trained on small scale dataset and show the performance difference experimentally.

An important aspect to highlight is that the formulation of questions, paragraphs and answers from both *SQUAD v1.0* dataset and *BIOASQ* dataset are very similar. Only the vocabulary and paragraph lengths are different (*BIOASQ* dataset has shorter paragraph lengths in gold standard data). The task definition provided by the organizers which states that all the paragraphs are useful to answer the questions, enables to use a *Reading Comprehension* model.

Datasets	Train	Dev	Test
BIOASQ 4b	427	59	161
BIOASQ 5b	544	75	150
BIOASQ 6b	685	94	161
SQUAD v1.0	87,599	10,570	9,533
QUASAR-T	37,012	3,000	3,000

Tab. 4.1: Datasets used in the experiments along with their splits. The numbers represent number of questions.

At the time of performing these experiments, *DRQA* model by (Chen, Fisch, et al., 2017) was ranking on top on the SQUAD leaderboard with the source code released by the authors[9]. The model is built for the *Reading Comprehension* task i.e. the model would take as input - a question and a paragraph, and output an answer span in the paragraph. We use this model and modify the dataset to fit the format.

Data modification

We modified the BIOASQ dataset in order to fit it into *Reading Comprehension* task style data by considering only those paragraphs which contained the gold standard answer and ignoring the rest. We do this the same way as (Wiese et al., 2017b) by performing distant supervision considering the gold standard answers and paragraphs.

An example from SQUAD dataset and BIOASQ dataset:

Question: The atomic number of the periodic table for oxygen?
Paragraph: Oxygen is a chemical element with symbol O and atomic number **8**. It

[9] https://github.com/facebookresearch/DrQA

is a member of the chalcogen group on the periodic table and is a highly reactive nonmetal and oxidizing agent that readily forms compounds (notably oxides) with most elements.
Answer: 8

Question: Which topoisomerase is essential in yeast?
Paragraph 1: Yeast DNA topoisomerase II is encoded by a single-copy, essential gene.
Paragraph 2: Topo II performs topological modifications on double-stranded DNA molecules that are essential for chromosome condensation, resolution, and segregation.
Answers: Topoisomerase II, Topo II

After these modifications, we can highlight the differences between SQUAD dataset and BIOASQ datasets:

- BIOASQ can have multiple paragraphs and SQUAD has 1 paragraph.
- SQUAD paragraph always contains an answer and BIOASQ paragraphs might not always contain exact matching gold answers.
- SQUAD has 1 answer, BIOASQ has multiple answers (as shown in the above example).

DRQA considers each question and a paragraph as a pair for each sample datapoint. It does not take into account multiple paragraphs per question or multiple answers per paragraph. Therefore in order to modify our data to match this format, we repeat the same question with multiple paragraphs creating same number of samples as the number of paragraphs and consider only one gold standard answer per paragraph. We keep only the positive paragraphs (paragraphs containing an answer) in the dataset.

The *DRQA* model is originally trained and tested on SQUAD v1.0 dataset. We use the same model trained on SQUAD dataset, and perform domain adaptation by fine-tuning the model on our modified BIOASQ dataset. Figure 4.6 shows the procedure we follow.

The *Open domain model* is the *DRQA* trained on SQUAD v1.0 with best performing parameters of the model (best scores on validation set). The *Biomedical + Open*

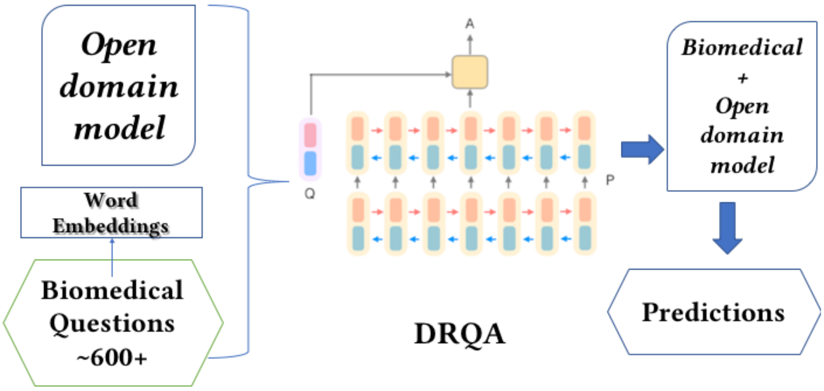


Fig. 4.6: Domain Adaptation from open domain to biomedical domain using DRQA model

domain model is the same best performing *Open domain model* trained again on modified BIOASQ dataset. This model is used to predict answers on test sets of BIOASQ. Both the processes use the same Glove word embeddings.

For BIOASQ evaluation, if we have only 1 paragraph relevant for a question, we take 5 predictions from the same paragraph as Top-5 answers and rank it based on the decreasing score of probability. Top-1 probability answer is evaluated for the strict accuracy as per official evaluation. And Top-5 is evaluated for lenient accuracy. If we have more than 1 paragraphs per question, we take at least 1 prediction (best candidate prediction with highest probability) per paragraph until we have 5 predictions. Further all the predictions are ranked based on the decreasing score of their answer probabilities.

4.2.3 Importance of Pre-Training and Fine-Tuning

To show the importance of *Pre-Training* and *Fine-Tuning* for domain adaptation to biomedical domain, and also show the necessity of doing this to be able to use deep learning methods efficiently, we experimented three approaches on a single model *DRQA* without altering any hyperparameters. Default parameters as those used by (Chen, Fisch, et al., 2017) in their code[10].

1) *No-Pre* model is the *DRQA* model trained on BIOASQ dataset only - *Deep learning on small scale data.*

2) *No-Fine* model is the *DRQA* model trained on SQUAD v1.0 dataset only. - *Deep learning on large scale data.*

[10] https://github.com/facebookresearch/DrQA

Datasets	No. of Ques	Metrics	No-Pre	No-Fine	Pre+Fine
BIOASQ 4b	161	S.Acc	08.98	23.96	**24.00**
		L.Acc	16.56	35.26	**39.21**
		MRR	11.36	28.40	**29.34**
BIOASQ 5b	150	S.Acc	25.91	32.17	**32.43**
		L.Acc	34.86	45.58	**47.73**
		MRR	29.27	36.75	**38.37**
BIOASQ 6b	161	S.Acc	13.40	26.24	**26.72**
		L.Acc	27.08	40.60	**43.72**
		MRR	19.20	32.57	**33.80**
Average		S.Acc	16.09	27.45	**27.71**
		L.Acc	26.16	40.48	**43.55**
		MRR	19.94	32.57	**33.83**

Tab. 4.2: Results reporting the importance of Pre-Training and Fine-Tuning a model. All scores are computed on the official test sets combined into one. S.Acc is Strict Accuracy, L.Acc is Lenient Accuracy (the correct answer is in the top 5) and MRR is the Mean Reciprocal Rank for the correct answer in Top-5 answers. *No-Pre* is for *No-Pretraining* on open domain, *No-Fine* is for *No-Finetuning* on biomedical domain and *Pre+Fine* is for *Pretraining and Finetuning* on open domain and biomedical domain.

3) *Pre+Fine* is the *DRQA* model trained on SQUAD v1.0 dataset and fine-tuned on BIOASQ dataset - **Domain Adaptation**.

Results are shown in the Table 4.2 for different BIOASQ test sets. When a model is only trained on a small dataset like BIOASQ, the results are very low as shown in the column *No-Pre*, because deep neural network model do not perform the best when the source dataset is small scale. When a model is trained on a large dataset like SQUAD v1.0, the model can be straight away used to predict results on the biomedical dataset. *No-Fine* shows a clear improvement doing so, against *No-Pre*, because of the large scale dataset it is trained on.

Lastly, *Pre+Fine* is the model which underwent pre-training on a large scale dataset and fine-tuning on the biomedical dataset which clearly shows an improvement over the other approaches as the model which is well learnt on a large scale dataset is tuned to fit a small scale dataset. This set of experiments show that the best approach is to do pre-training on large scale datasets and fine-tuning on small scale datasets, which is a feasible approach to use deep learning models on small scale datasets.

The difference between *No-Fine* and *Pre+Fine* shows the need for domain adaptation. The *No-Fine* model is a good model with good performance because it has been trained on a large scale dataset, but it can be further improved by doing the above process.

4.2.4 Comparison of word embeddings for domain adaptation

While performing domain adaptation, one hurdle we came across was the set of missing vocabulary terms across different domain word embeddings. The input word embeddings for both *Pre-training* and *Fine-tuning* process should be the same for the overlapping words. Therefore a word having one word vector in word embedding space cannot be different while doing pre-training or fine-tuning. Which means a common word embedding space is required for this purpose.

An easy way is to keep the same word embeddings space for both *Pre-training* and *Fine-tuning* processes. This gives raise to several questions. *Using both domain data together is better? or one is better than the other?* , *Which algorithm is the best for this purpose?*.

In order to check with different domain word embeddings, different algorithms and different hyperparameters required to train them, we train different word embeddings with CBOW and Skipgram models of Word2Vec by (Mikolov et al., 2013) with different hyperparameters and different data from open domain and biomedical domain.

For training biomedical domain word embeddings we chose the BIOASQ 5A task data which consist of 12,834,585 PUBMED articles as an input corpus. We preprocessed this dataset to remove special characters and use the Gensim tool[11] to train word embeddings with 50, 100, 200, 300, 400 dimensions with CBOW and Skipgram algorithms. We also use Global Vectors (Glove) which was trained on 840B tokens, 300 dimensions and available to download freely[12] to compare performance. We also combine open domain and biomedical domain data into one big corpus and use it to train word embeddings using Word2vec.

We do an extrinsic evaluation of word embeddings by using them on the downstream task of question answering. BIOASQ Task B contains five different test batches with distinct question sets. We retrained the *DRQA* model on BIOASQ 2017 5B training data after removing each test set.

Table 4.3 presents the comparison of five word embedding spaces tested on the five test sets (Test-1 to Test-5) and a set with all test sets combined (All). *BIOASQ 4* with 200D performed worse on our experiments. These embeddings are given by the organizers of BIOASQ task that is available on their website. *BIOASQ 5* embeddings

[11] https://radimrehurek.com/gensim/models/word2vec.html
[12] https://nlp.stanford.edu/projects/glove/

	Nb. of Ques.	BIOASQ 4 |V| = 1.7M |T| = 2.2B	BIOASQ 5 |V| = 2.1M |T| = 2.3B	Wikipedia |V| = 2.13M |T| = 2.19B	Wiki+BIOASQ 5 |V| = 4M |T| = 4.49B	Glove |V| = 2.2M |T| = 840B
Test-1	39	33.33	46.15	46.15	51.28	**58.97**
Test-2	31	35.48	48.39	45.16	48.38	**51.61**
Test-3	26	38.46	**65.38**	**65.38**	**65.38**	61.53
Test-4	31	35.48	45.16	**51.61**	41.93	41.93
Test-5	33	45.45	57.57	60.61	60.61	**66.66**
Average	–	37.64	52.53	53.78	53.52	**56.14**
All	160	33.12	51.25	45.0	50.0	**52.5**

Tab. 4.3: Accuracy (top 5) on 4B test with different Embeddings: |V| = vocab, |T| = token counts

were trained by us with a Skipgram model and 300D. We can see that although *Glove* embeddings are trained on Web Crawl data and not specifically biomedical data, it performs better than the rest trained on biomedical data because of the large training data of Glove. *Wiki+BIOASQ 5* trained with Skipgram and 300D on data of *BIOASQ 5* and Wikipedia articles, has second best accuracy after *Glove* because of the domain specific training data even though it is smaller compared to Glove's training data.

	CBOW		Skipgram	
Dims	Strict	Lenient	Strict	Lenient
100	30.62	**48.75**	31.25	50.0
200	28.75	47.5	**33.75**	50.0
300	**31.87**	**48.75**	31.87	**51.25**
400	28.75	46.87	30.0	48.75

Tab. 4.4: Comparision of Word2vec models on 4B Test set (Testset "All" from Table 4.3)

Table 4.4 presents a comparison of different embedding spaces trained on different dimensions (namely 100, 200, 300, 400) with CBOW and Skipgram models as described in (Mikolov et al., 2013), where the performance is calculated based on Strict Accuracy (Top-1) measure of BIOASQ 4B test sets. It is evident from the table that Skipgram performs better than CBOW when the dimensions are higher. But 300 dimensions is found to be optimal in terms of both strict and lenient accuracy. Increasing it to 400 dimensions did not fetch better results.

These experiments highlight the importance of choosing right word embeddings for biomedical domain QA system. Glove performs better on average because of large amount of data it is trained on, and pretraining on SQUAD which has a large set of open domain questions makes the pretrained QA model to learn better representations. Whereas the biomedical embeddings are trained on lesser data and domain specific vocabulary which has a negative impact over the pretraining of SQUAD. Therefore we further use Glove embeddings for all our experiments as it showed better performance in the above experiments.

In the above set of experiments, we consider only positive paragraphs (paragraphs containing an answer) in the datasets for answer extraction which suits the definition of *Reading Comprehension* task. BIOASQ datasets consists of some negative paragraphs (paragraphs that do not contain an answer) as well. Therefore in the next section we experiment by modelling the task as an *OpenQA* task for pre-training.

4.3 Domain adaptation with Different Models and Data

In the previous section we focussed on how domain adaptation can be done using a *Reading Comprehension* task model *DRQA* with modifying different word embedding inputs. In this section we explore two other aspects of performing domain adaptation. 1) *Different QA models*. 2) *Different Large Scale Datasets*.

Our hypothesis is that the pre-training data and different modelling can also result in performance variations. The following sections show which models and which datasets are better for domain adaptation towards biomedical domain.

4.3.1 Comparison of Different Pre-Training Models

In section 4.2.2 we already discussed about how a *Reading Comprehension* model *DRQA* is used for domain adaptation. We now consider another question answering task named *Open QA*. *Open QA* is a QA task where a question is given and the goal is to retrieve an answer. Since this is an open task, answer can be retrieved either from textual sources or knowledge graphs or ontologies. In our work we focus only on textual sources. An answer has to be retrieved from a set of documents or passages of textual sources as Wikipedia articles or news or scientific articles. Answers are also usually short phrases or entities.

In deep neural network approaches for Open QA, generally answers are extracted using a reading comprehension model on the subset of the retrieved documents or passages considered as relevant (Dhingra, Mazaitis, and William W. Cohen, 2017b; Joshi et al., 2017).

One of the main differences between *Reading Comprehension (RC)* and *Open QA* tasks is that the answer must be present in the paragraphs (or documents) for *Reading Comprehension (RC)*, but for *Open QA* this condition might not hold true because the retrieved documents considered to be relevant to the question might not contain any answer. Multiple paragraphs can contain the answer as well. An effective *Open QA* model must consider all these into account.

Below is an example from the dataset.

Q: Which calcium channels does ethosuximide target?
A: T-type calcium channels
P1: ..neuropathic pain is blocked by ethosuximide, known to block T-type calcium channels,..
P2: Theta rhythms remained disrupted during a subsequent week of withdrawal but were restored with the T-type channel blocker ethosuximide.

However, as shown in the example one paragraph (P1) has the gold standard answer and the other (P2) does not (i.e. it does not contain the exact match of the answer string). Therefore this resembles more like an *Open QA* task than a *Reading Comprehension* task because of several reasons. 1) The existence of paragraphs without answers even though they are considered relevant. 2) Multiple paragraphs containing the same answer. Therefore we experiment by pre-training and fine-tuning in three different ways for BIOASQ task by taking into account the above two considerations.

In its general definition, the *OpenQA* task contains questions and their short answers without any given paragraphs. BIOASQ organizers already provide paragraphs in the gold standard data. Therefore, *OpenQA* can be formulated as a parent task which involves two child tasks, 1) Ranking the relevant paragraphs for a question and 2) Extracting a short answer from the paragraphs. In Open QA, the first task is generally referred as paragraph selection or answer sentence selection and the second task is often modelled as Reading Comprehension although there exists several correct and incorrect paragraphs. Open QA models should distinguish if the paragraph is correct and then extract the answer unlike the RC models.

We use the two models for *RC* and *Open QA* which are shown in the Figure 4.7.

Open QA model described in section 4.1.3 is used for the implementation. While adapting the model to BIOASQ, the number of answers to be extracted for BIOASQ is Top-5 and not Top-1. Because of this, instead of choosing from only the top most probable paragraph, we select top 5 answers from combined probability scores in equation 4.20, which might consider 1 or more paragraphs to extract answers from.

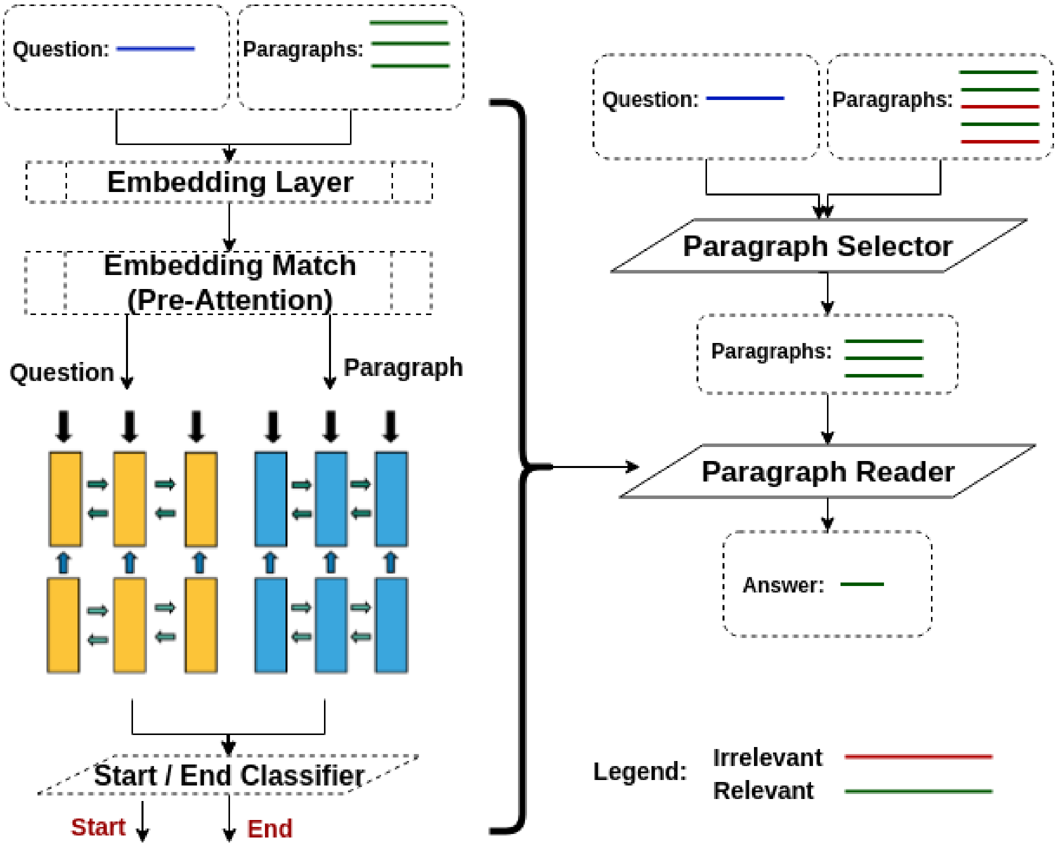


Fig. 4.7: Left: DRQA - Paragraph Reader (*RC task*). Right: PSPR - Paragraph Selector and Paragraph Reader model (*Open QA task*)

$$Pr(a|q, P) = \sum_{p_i \in P} Pr\left(a|q, p_i\right) Pr\left(p_i|q, P\right) \tag{4.20}$$

In this set of experiments, we apply *Type 1* domain adaptation.

The data for pre-training

Two datasets correspond to each of the two tasks: SQUAD V1.0 dataset for *RC* task and QUASAR-T dataset for *Open QA* task and we show below their differences.

- QUASAR-T which is based on Trivia questions is generated synthetically, and SQUAD is annotated manually by humans on a crowd-sourcing platform.

- Each question in QUASAR-T is associated to 100 sentence-level passages retrieved from ClueWeb09 dataset based on Lucene, whereas SQUAD 1.0 has 1 relevant paragraph from Wikipedia.

- Some paragraphs in QUASAR-T do not have an answer. [13]

Comparing the above differences with BIOASQ dataset, the QUASAR-T dataset resembles more closely to BIOASQ than that of SQUAD v1.0 due to the following reasons.

- BIOASQ data has more than 1 relevant paragraphs per question.

- Some paragraphs do not have an answer.

Experiments with Gold standard paragraphs

For studying the modelling of the BIOASQ QA task as a *Reading Comprehension* task, we use SQUAD v1.0 dataset for pre-training and experiment with the *DRQA* model as explained in the previous section. For studying its modelling as an *Open QA* task, we use QUASAR-T dataset for pre-training and experiment with *PSPR* model.

As the *PSPR* model is a cascaded model with paragraph selector and paragraph reader, we use the paragraph probabilities predicted by the paragraph selector and multiply them with the answer probabilities obtained using *DRQA* model to select the Top-5 answers which have combined higher probabilities. The *PSPR* model has a selector model which predicts a probability score for each paragraph which signifies the answer presence. We use these paragraph probability scores from *PSPR* model and the answer probability scores from *DRQA* model and multiply them to choose the answer candidates. The results for this experiment is highlighted in Table 4.5 as *DRQA+PS*. This is different *PSPR* model because *PSPR* learns the model by combining probabilities, whereas *DRQA+PS* does not learn but just use output probability scores.

Results are shown in Table 4.5 for different BIOASQ test sets. We compare different model results with BioBert scores reported in (J. Lee et al., 2019). The scores from *PSPR* model shows the performance on Strict and Lenient accuracy on 4b, 5b and 6b test sets. By taking paragraph probability into account *PSPR* allows to better rank Top-1 correct answer than BioBert which extracts answers from all paragraphs and

[13] SQUAD 2.0 is a variant of SQUAD dataset which contains questions without answers. We do not use this because the reference models also do not use v2.0 to pre-train.

chooses the maximum probability scored answer only based on answer probability. Although *PSPR* has a reader model similar to *DRQA*, considering the paragraph probability improves the answer extraction in *PSPR* model.

Datasets	Metrics	BioBert by (J. Lee et al., 2019)	DRQA	DRQA+PS	PSPR
BIOASQ 4b	S.Acc	**36.48**	24.00	26.22	30.28
	L.Acc	**48.89**	39.21	32.33	40.34
	MRR	**41.05**	29.34	26.54	34.19
BIOASQ 5b	S.Acc	41.56	32.43	30.62	**46.59**
	L.Acc	**54.00**	47.73	47.86	53.76
	MRR	46.32	38.37	36.96	**49.55**
BIOASQ 6b	S.Acc	35.58	26.72	26.50	**43.91**
	L.Acc	**51.39**	43.72	42.16	51.34
	MRR	42.51	33.80	32.07	**45.70**
Average	S.Acc	37.87	27.71	27.78	**40.26**
	L.Acc	**51.43**	43.55	40.78	48.48
	MRR	**43.29**	33.83	31.85	43.14

Tab. 4.5: *DRQA* is the Reading Comprehension model by (Chen, Fisch, et al., 2017), *PSPR* is the Open QA model by (Y. Lin et al., 2018), *DRQA+PS* is answers chosen with scores by multiplying answer probabilities of DRQA with Paragraph Selector probabilities of PSPR. SOTA scores are reported by (J. Lee et al., 2019) who average the best scores from each batch (possibly from multiple different models). Results from BIOASQ 4b, 5b and 6b test sets. 7b test set cannot be evaluated yet due to lack of gold standard answers. S.Acc is Strict Accuracy, L.Acc is Lenient Accuracy and MRR is Mean Reciprocal Rank. Experiments are done with the original BIOASQ data.

Experiments with Longer paragraphs (modified BIOASQ data)

For the BIOASQ task we noted that the method used by (J. Lee et al., 2019) with BioBert modified the original paragraphs. For computing the BioBert model, the authors have retrained the original BERT model (Devlin et al., 2018) using Pubmed and PMC articles. For applying it on the BIOASQ task, the authors use longer documents (instead of the actual snippets) from Pubmed corresponding to the data given by BIOASQ in the "documents" field to access the Pubmed documents for each question. Therefore this modification of the gold standard dataset leads to different results for BioBert compared to the gold standard data performance.

The exact pre-processing of BIOASQ dataset in order to do this is detailed in (Yoon et al., 2019). The authors have used different strategies and found that using full abstracts from the pubmed document is more useful than using the BIOASQ gold standard snippet which is a small part of the abstract. The authors released the data[14] with full abstracts which we use in some of our experiments for comparison.

Similar to these modifications, the authors of (Hosein et al., 2019) perform modifications of the BIOASQ gold standard data by using pubmed abstracts as paragraphs instead of the given gold standard paragraphs (which are part of the of the pubmed abstracts). The authors also use the paragraphs obtained from a system which participated in BIOASQ paragraph retrieval task.

They found that the performance is better when the full abstract text is used instead of gold standard data in some cases, and gold standard training data worked better in some cases. The authors conclude that the paragraph retrieval also plays major role than the QA model itself and report that the domain adaptation is not always useful as the non fine-tuned model performed better in some cases. We do not compare our scores with their work because the authors pre-train their models on Google Natural Questions dataset (Kwiatkowski et al., 2019) which contains atleast 3 times more data than SQUAD dataset and diverse set of questions obtained from Google search logs. Because of this pre-training difference with a different dataset, the results are not directly comparable with the systems which have pre-trained on SQUAD dataset. We discuss more about this in the next section.

In order to evaluate the importance of this data modification, we did two experiments and also report results on BioBert with modified paragraph data from (J. Lee et al., 2019) : 1) DRQA with longer contexts 2) BioBert with unaltered data from BIOASQ.

The results are shown in Table 4.6. The results of *BioBert* is as presented in (J. Lee et al., 2019) where the authors have fine-tuned the models first using SQUAD v1.0 dataset and adapted it to BIOASQ data. We use the modified dataset to experiment it with the *DRQA* model to determine if it would improve the performance of the pre+fine *DRQA* model as reported in Table 4.2. We got lower performances to that of the *DRQA* model trained on the original BIOASQ data.

For comparison, we try the BioBert model on the original BIOASQ data i.e. paragraphs given by BIOASQ data and not pre-processed. The results in Table 4.6, under the column *BioBert-Unaltered* represents these results. It is evident that the modification performed on the BIOASQ data fetches better results using BioBert model.

[14] https://github.com/naver/BioBert-pretrained

Datasets	Metrics	SOTA	DRQA	BioBert-Unaltered	BioBert by (J. Lee et al., 2019)
BIOASQ 4b	S.Acc	20.59	18.49	13.08	**36.48**
	L.Acc	29.24	32.51	18.54	**48.89**
	MRR	24.04	23.88	15.48	**41.05**
BIOASQ 5b	S.Acc	**41.82**	28.92	22.84	41.56
	L.Acc	57.43	46.54	32.46	**54.00**
	MRR	**47.73**	35.88	25.94	46.32
BIOASQ 6b	S.Acc	25.12	21.70	16.35	**35.58**
	L.Acc	40.20	41.51	22.61	**51.39**
	MRR	29.28	28.60	18.72	**42.51**
Average	S.Acc	29.18	23.03	17.42	**37.87**
	L.Acc	42.29	40.18	24.53	**51.43**
	MRR	33.68	29.45	20.04	**43.29**

Tab. 4.6: Experiments with data containing longer contexts (Document level) by (J. Lee et al., 2019). *DRQA* is a Reading Comprehension model by (Chen, Fisch, et al., 2017). *BioBert-Unaltered* is the original BIOASQ dataset with questions and paragraphs which contain answers. *BioBert by (J. Lee et al., 2019)* is the modified BIOASQ dataset where the paragraphs are longer paragraphs (documents from respective articles), where all the models are pre-trained on SQUAD v1.0 dataset and finetuned on BIOASQ dataset. SOTA scores are reported by (J. Lee et al., 2019) who average the best scores from each batch (possibly from multiple different models). Results from BIOASQ 4b, 5b and 6b test sets. 7b test set cannot be evaluated yet due to lack of gold standard answers. S.Acc is Strict Accuracy, L.Acc is Lenient Accuracy and MRR is Mean Reciprocal Rank.

Conclusion

In this set of experiment we compare two QA models based on i.e. 1) Reading Comprehension task 2) Open QA task, and found that the performance is better when using an Open QA model than a Reading Comprehension model. We report the performance on different datasets and show that in some cases *OpenQA* modelling outperforms the state-of-the-art systems of BIOASQ which use domain adaptation using *Reading Comprehension* model (Wiese et al., 2017a; J. Lee et al., 2019) in average.

Based on a different pre-processing done by (J. Lee et al., 2019) on the biomedical dataset by using longer contexts from documents than shorter contexts, we found that the Reading Comprehension model performs worse on the pre-processed longer contexts (which are longer documents than paragraphs) compared to the shorter contexts (paragraphs) originally given by BIOASQ data. On the other end, a large pre-trained language model such as BERT performs much better on the pre-processed longer contexts than shorter contexts.

4.3.2 Comparison of Pre-Training with Different Datasets

In the previous section, the importance of *Pre-training* and *fine-tuning* a.k.a *Domain adaptation* can be seen on small scale datasets in order to use deep neural network models effectively. We also state that the deep learning models perform better when trained on large scale datasets. This assumption gives raise to several concerns to address before using different datasets to train deep neural network models.

Some of those concerns are:

1. How *large* is large enough for a dataset?
2. What is the minimum size required for deep neural network models to learn efficiently?
3. What kind of data should be used?
4. Are synthetic datasets better than human annotated ones?
5. How do we choose the best dataset for pre-training?

To address some of the above concerns in terms of question answering and deep learning methods, we present some experiments in this section which uses a single model namely BERT-Base (Devlin et al., 2018). BERT[15] has two models 1) BERT-Base and 2) BERT-Large. BERT-Large takes approximately 12-13 hours to fine-tune on SQUAD dataset on one GPU, whereas BERT-Base takes around 2 hours of time. Therefore we decided to use BERT-Base for our experiments. The BioBert model which we have used in other set of experiments on other datasets is pre-trained on the BERT-Base model. The goal is to experimentally understand the performance variation of QA models trained on different datasets and compare the performance variation while using it for domain adaptation.

This study was inspired by the work of (Talmor and Berant, 2019) who perform an empirical investigation of generalization of QA datasets across different other datasets. The authors use two models 1) Bi-LSTM based attention model - DocQA 2) BERT Base for QA and perform several experiments by training models on one dataset and testing it on another. They also do fine-tuning on target datasets (not always small scale) and show that the fine-tuning does have positive impact.

[15] https://github.com/google-research/bert

One of the conclusions from their study is that the QA models are often overfitting to source datasets and their own data distributions and not well suited when data comes from a different source or is of a different type. Therefore the authors create a single dataset called MultiQA by using several QA datasets listed in Table 4.8 where some percentage of questions were randomly picked from all datasets used in the experiments to create the MultiQA dataset which fetches the best results over all the test sets of the datasets used on the *Reading Comprehension* task, using the same model BERT-Base. Some important observations from this work are relevant to our work:

- A model trained on sufficiently large scale data is good at performance for generalization.
- An ideal dataset to obtain *The Best* performing QA model involves mixing up questions from different datasets which are created using different sources of data and different annotation schemes.
- Generalization improves when the model learns information from the target distribution of data.

Based on some observations and conclusions mentioned in the above study, we decided to perform a similar study on domain adaptation by using a single model pre-trained with different datasets to measure the performance of fine-tuning process towards biomedical domain.

Our intuition is that, a model with exact same hyperparameters perform differently on a downstream task when pre-trained with different datasets. Mainly because of the nature and the scale of the dataset. Human annotated data may perform better than synthetic ones which is why human annotated datasets are widely used for benchmarking QA models. Our goal is to experimentally determine these aspects on biomedical domain adaptation.

Experiments

We pre-train the BERT Base model on datasets from Table 4.8 individually and fine-tune the model to BIOASQ datasets from Table 4.7 individually.

For the datasets in table 4.8, note that the *SQUAD 2.0*, *HotpotQA*, *NewsQA*, *SearchQA* and *TriviaQA* are not exclusively for *Reading Comprehension* task as some datasets have relevant and irrelevant snippets like the datasets for *OpenQA* task. Therefore,

Datasets	Train	Dev	Test
BIOASQ 4b	427	59	161
BIOASQ 5b	544	75	150
BIOASQ 6b	685	94	161

Tab. 4.7: Small scale BIOASQ datasets used in the experiments for fine-training, with their splits. The numbers represent number of questions.

Datasets	Train	Dev	Test
SQUAD v1.0	87,599	10,570	9,533
SQUAD v2.0	130,319	11,873	8,862
Hotpot QA	90,564	7,405	7,405
News QA	107,673	5,988	5,971

Tab. 4.8: Large scale datasets used in the experiments for pre-training, with their splits. The numbers represent number of questions.

as done by the authors of (Talmor and Berant, 2019), we also pre-process and keep only those questions whose paragraphs contain the answers. If there are irrelevant and relevant paragraphs in the datasets, we only keep relevant paragraphs and questions . This allows to use SQUAD v1.0 models with these datasets.

For the following experiments we use the single model and the same set of hyper parameters for all the experiments as used in the code[16] for RC task. Table 4.9 shows experimental results on BIOASQ 4,5 & 6 task test sets.

The experiments are performed on the models trained on open domain large scale datasets from Table 4.8 and fine-tuned on the BIOASQ data. We have two special datasets which are combined datasets created using several other datasets into a single large dataset. 1) SQUAD v1.0, SQUAD v2.0, NewsQA, HotpotQA. 2) SQUAD v1.0, SQUAD v2.0. The above datasets are created by combining the datasets mentioned into a big collection by adding 100% of data from each set. Due to lack of time we did not experiment more combinations of these with varying amount of data from different sets. We compare our results with BioBert (Yoon et al., 2019) whose work scored the best at BIOASQ 7. Since we do not have access to BIOASQ 7 gold standard data for the test sets, we cannot evaluate our models on the test set.

SQUAD v2.0 dataset pre-training seems to clearly outperform *SQUAD v1.0* on average on Strict Accuracy and MRR, which almost all previous works such as (Wiese et al., 2017b; J. Lee et al., 2019) have used for pre-training their models.

[16] https://github.com/huggingface/transformers

Index	Datasets	Measures	BIOASQ 4	BIOASQ 5	BIOASQ 6	Average
1	NewsQA	Strict	30.33	44.58	37.21	37.37
		Lenient	46.68	57.42	52.87	52.32
		MRR	36.74	49.82	43.54	43.36
2	HotpotQA	Strict	31.61	36.72	33.58	33.97
		Lenient	45.81	52.14	53.74	50.56
		MRR	37.07	42.55	41.97	40.53
3	SQUAD v1.0	Strict	30.30	42.12	36.73	36.38
		Lenient	52.66	56.99	54.60	54.75
		MRR	39.51	48.16	43.83	43.83
4	SQUAD v2.0	Strict	34.00	42.91	41.01	**39.30**
		Lenient	50.50	57.28	56.49	54.75
		MRR	**41.30**	48.02	**47.17**	**45.49**
5	SQUAD v2.0 & v1.0	Strict	31.13	**46.53**	38.90	38.85
		Lenient	48.24	56.47	54.88	53.19
		MRR	37.92	**50.11**	44.78	44.27
6	SQUAD v2.0 & v1.0 HotpotQA, NewsQA	Strict	**34.70**	39.36	**41.41**	38.49
		Lenient	51.40	57.69	56.64	55.24
		MRR	41.27	46.76	47.13	45.05
7	BioBert at BIOASQ	Strict	28.57	44.00	42.86	38.47
		Lenient	47.82	56.67	57.14	53.87
		MRR	35.17	49.38	48.41	44.32

Tab. 4.9: Experiments with different single datasets with fine-tuning on BIOASQ data. Results averaged over 5 official test sets. The paragraphs used in these experiments has been obtained from the authors of BioBert (J. Lee et al., 2019) which is a modified long document data and not BIOASQ gold standard paragraphs. Best scores are highlighted in Bold according to the different BIOASQ tasks and Average.

For individual BIOASQ (BIOASQ 4, 5 and 6) evaluations, on BIOASQ 4 the dataset *(SQUAD v1.0, SQUAD v2.0, NewsQA, HotpotQA)* fetches better scores for Strict accuracy and *SQUAD v2.0* fetches better scores for MRR. On BIOASQ 5 the dataset *(SQUAD v2.0, SQUAD v1.0)* fetches better scores for both Strict accuracy and MRR. On BIOASQ 6 the dataset *(SQUAD v1.0, SQUAD v2.0, NewsQA, HotpotQA)* fetches better scores for Strict accuracy and *SQUAD v2.0* fetches better scores for MRR.

On average, the scores of *SQUAD v2.0* dataset are better than the BioBert (Yoon et al., 2019) scores which are pretrained on *SQUAD v1.0* dataset. In the next set of experiments we experiment several BERT models including BioBert on the *SQUAD v2.0* dataset.

There are multiple BERT models popular these days such as Roberta by (Y. Liu et al., 2019), XLNet by (Zhilin Yang, Dai, et al., 2019), DistilBERT by (Sanh et al., 2019) etc. Some are trained using different tweaks in the architectures and training methods, and some are trained on different datasets. We are interested in testing

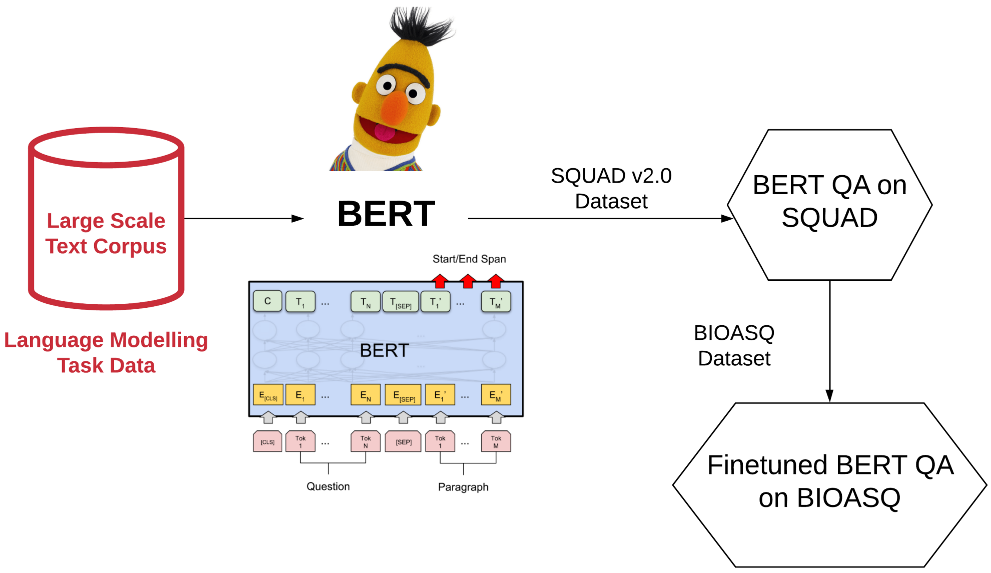


Fig. 4.8: BERT model training process from Language modelling task to BIOASQ QA task.

BERT trained on different datasets like BioBert by (J. Lee et al., 2019) which was shown to be useful for several biomedical tasks by just fine-tuning the original BERT model on some large scale biomedical text corpus.

The Figure 4.8 shows a pipeline of training tasks done on the BERT model to obtain a BIOASQ QA model. In this experiment we use BERT-BASE models which have modified only the *Large Scale Text Corpus* data for the language modelling task as highlighted in red colour in the Figure 4.8. The underlying BERT model (BERT-BASE), the QA task pre-training (SQUAD v2.0) and fine-tuning (BIOASQ) datasets remain the same.

We believe that the pre-training language modelling task of BERT on different datasets, has a greater impact on the performance of BERT on downstream tasks like QA task.

We use three BERT models trained on different domain data.

1. BERT Base Google by (Devlin et al., 2018) - the original BERT model with Base (smaller in size) trained on large scale open datasets.

2. BioBert by (J. Lee et al., 2019) - BERT model trained on original google data and later finetuned on PubMed 200K articles + PMC 270K articles.

3. Scibert by (Beltagy et al., 2019) - BERT model trained on multi-domain corpus of scientific publications.

Datasets	Measure	BIOASQ 4	BIOASQ 5	BIOASQ 6	Average
BERT Base (Google)	Strict	22.02	24.22	19.50	21.91
	Lenient	33.16	38.69	39.31	37.05
	MRR	26.37	29.65	26.69	27.57
BioBert	Strict	**34.00**	**42.91**	**41.01**	**39.30**
	Lenient	**50.50**	**57.28**	**56.49**	**54.75**
	MRR	**41.30**	**48.02**	**47.17**	**45.49**
SciBert	Strict	28.20	32.83	33.55	31.52
	Lenient	43.35	43.39	46.54	44.42
	MRR	34.07	37.02	38.66	36.58

Tab. 4.10: Experiments with BERT-BASE models trained on different text corpus for Language Modelling tasks. We pre-train these models on SQUAD V2.0 dataset and then fine-tune on BIOASQ dataset. Results averaged over 5 official test sets.

Table 4.10 shows experimental results on BIOASQ 4, 5 & 6 task datasets on different BERT-BASE models. We use the SQUAD v2.0 dataset for pre-training as shown above in table 4.9 that this dataset is the better option to pre-train a BERT model and fine-tune to BIOASQ than the rest (excluding the combination datasets).

Table 4.10 shows that the BioBert model by (J. Lee et al., 2019) is the best choice as shown earlier in their article. Even though Scibert is trained on scientific domain data, it is not the best choice for biomedical tasks as seen in the results above. Google BERT performed with least scores as it is not trained specifically on any biomedical corpus.

In the above set of experiments, we try different datasets on BERT models and experimentally verify that the SQUAD v2.0 is the better single dataset to pre-train models than SQUAD v1.0 maybe because of the presence of paragraphs without answers. And BioBert is the best suited pre-trained domain data for BERT model on biomedical tasks. While SQUAD v2.0 performs the best among single dataset pretraining, the combination dataset which combines 4 datasets (SQUAD v1.0, SQUAD v2.0, NewsQA, HotpotQA) is the best option to fetch better results.

In the work of (Hosein et al., 2019), the authors pre-train Google BERT on Natural Questions dataset by (Kwiatkowski et al., 2019) and find that in some cases, the results are better than BioBert. We believe that this phenomenon would be different for the authors if they had trained their system with SQUAD dataset and not Natural Questions (NQ) as the NQ dataset has 3 times larger than SQUAD and contains diverse range of questions. Our results and findings in this section supports our argument that pre-training QA datasets also play a major role in downstream QA system performance.

Also since the above experiments were conducted on BERT models which are trained on large scale data for language modelling, the performance gain from one dataset to another might be subtle. It would be interesting to see how traditional deep learning models like CNN or RNN based models like *DRQA* and *PSPR* would perform when pre-trained on different datasets. Note that training *DRQA* model takes approximately 4 hours, whereas fine-tuning BERT on SQUAD dataset takes around 2 hours which is why we chose BERT-Base for our experiments.

4.4 Conclusion

Using deep learning models directly on the small scale datasets (especially on different speciality domains like biomedical domain) will not fetch optimal results. In this chapter we addressed one of our research questions on building models which work both on small scale and large scale datasets to obtain better performance. Choosing right models which works better on open domain and adapting it to biomedical domain was the primary goal.

We presented and detailed three QA task models for *Reading Comprehension*, *Open QA* and *Answer Sentence Selection*. We also explained in brief about using BERT model for question answering. We presented some factors which we consider as important before choosing a model to work with and explained why we did some of our choices of models.

In this regard, we introduced the concept of domain adaptation from open domain to biomedical question answering and showed the impact on the performance gain. We also formally defined some terminologies used for domain adaptation and showed different ways of doing domain adaptation which is sometimes also referred to as *Transfer Learning* when the target task is different. The domain adaptation cannot performed be straight away because the data formats are different for the open domain dataset and the BIOASQ data, there we modified some aspects of the data and detailed about the process. This facilitates using state-of-the-art deep learning models for biomedical question answering.

We experimented these models with different word embeddings trained on open domain corpus, biomedical domain corpus and their mixture with various embeddings models like Word2vec, Glove etc. We concluded that using Glove embeddings was the best choice as it is trained on a large scale corpus.

While performing domain adaptation, choosing a good model is equally as important as choosing the right training datasets or right word embeddings. We studied 2 different QA task models suitable for biomedical domain adaptation. We concluded that the *Open QA* task outperforms the *Reading Comprehension* task for modelling

BIOASQ task. The model of (Y. Lin et al., 2018) for *OpenQA* task on BIOASQ outperforms BioBert model (Yoon et al., 2019) results on certain sets which shows that simpler models using Bi-LSTMs can be better than massively large scaled models like BERT on tasks like BIOASQ.

We experimented pre-training with different *Reading Comprehension* datasets by keeping the same model architecture and hyperparameters to show variability in the performance on downstream domain adaptation. SQUAD 2.0 dataset by (Rajpurkar, Jia, et al., 2018) was the best performing single dataset for pre-training the models. And a combination of 4 RC datasets for pre-training performed the best on the fine-tuning for BIOASQ dataset.

The neural network approaches usually focus on building end-to-end models and seldom focus on improving the predictions by post processing them. Generally, the objective of the neural models in QA is to score better on the Top-1 accuracy by learning the whole model on the QA dataset. Not a lot of emphasis is put on post-processing Top-K predictions. In the OpenQA task, a question is provided with several paragraphs some of which might contain the answer. Our hypothesis is that the semantic features from questions and paragraphs and structured information from different sources like UMLS can be used to further improve the performance of the OpenQA task. The predictions from the neural model are used to compute the features and input into a ML classifier which is modified as a ranker to better rank Top-1 predictions. The following chapter will focus on using different kinds of features in biomedical domain and open domain to improve the QA performance.

Our publications related to the work described in this chapter is listed below:

- **2019 - How to Pre-Train Your Model? Comparison of Different Pre-Training Models for Biomedical Question Answering.** - Sanjay Kamath, Brigitte Grau, Yue Ma. Proceedings of the 7th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering. ECMLPKDD, September 2019.

- **2018 - An Adaption of BIOASQ Question Answering dataset for Machine Reading systems by Manual Annotations of Answer Spans.** - Sanjay Kamath, Brigitte Grau, Yue Ma. Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering. EMNLP, October 2018.

5 Leveraging Structured and Semantic Information into Question Answering Models

In this chapter we address the topic of using existing data from different sources into the QA models to improve their performance. Ever since deep learning models have become prominent in the field of NLP, end-to-end models are on the rise which depend only on the input data to produce outputs. Our research concern in this regard is to use different ways to annotate, enrich and highlight certain aspects of the text data to improve the performance of these models without changing much of their underlying model.

When doing our experiments on BIOASQ data, we came across some issues for annotating the training data and automatically evaluating the results with gold answers. Using distant supervision with gold answers leads to omit a lot of answer variants. Thus, to overcome this problem, after studying its impact on the system results, we leverage biomedical entities and terminology present in the Metathesaurus UMLS[1] for improving the corpus annotation. These biomedical terms, entities and their types can be detected in free text data using Metamap. This tool gives an easy access to use UMLS by providing an interface to input text and obtain annotations about the biomedical entities, types and concept identifiers etc.

Neural network models do not explicitly use the semantic information such as Lexical and Expected Answer Types from questions which were used to improve prediction of answers in older models. The answer types are useful in highlighting the entity types in the paragraph text, which are likely to be the answers. We study different ways for modelling this information, by enriching the input data, or better representing entities by embeddings learned on semi-structured knowledge without changing the overall architecture of the system.

In order to exploit structured and semantic information sources for question answering in open domain and biomedical domain, we annotate, enrich and highlight the existing datasets with these extra information. We use it on different QA tasks (Answer sentence selection, Reading comprehension and Open question answering

[1] https://www.nlm.nih.gov/research/umls/index.html

tasks). All experiment results are compared with plain text results to show the performance variations.

Our third study concerns improving the system predictions in the open QA task by introducing a post-ranking process. The features used are based on the earlier mentioned expected answer types, and also on the redundancy of answers in several paragraphs. We conclude our study on the open QA task by experimenting state-of-the-art models on different and even newly built datasets, and highlight the remaining difficulty of this task when modelled by neural network approaches.

The organisation of this chapter is as follows:

- Annotation of Answer variants in BIOASQ dataset to highlight the actual performance of a Reading Comprehension model when the annotation is done extensively.

- We discuss about *Expected Answer Types* and *Lexical Answer Types* and their importance for question answering by performing a verification study with a QA model outputs.

- Annotation of entities in plain text w.r.t *Expected Answer Types* and using special entity embeddings for Reading Comprehension, Answer Sentence Selection and open domain question answering tasks.

- Improving the QA performance using semantic and structured information features such as lexical answer types and semantic types match from UMLS for biomedical data etc. for reranking answer candidates.

- Applying QA pipeline on different sub-task datasets to benchmark scores on Open Question Answering task.

5.1 Annotation of answer variants for BIOASQ dataset to improve performance of existing QA models in Biomedical domain

5.1.1 Introduction

"*What happens when a human (expert or non expert) annotated labelled dataset does not contain all possible variants of answer labels in their label set? Does that hurt a* *model's performance? What about true negatives while evaluating? Is the evaluation correctly done?*" - These are some of the questions which we wondered about when we came across BIOASQ dataset for the first time and built a system of QA on this dataset.

In fact, our results were pretty low compared to what we had obtained when manually did the evaluation ourselves (we are not biomedical experts, but we used reference gold standard answers to decide what is correct and what is not). The dataset of BIOASQ for question answering (task B phase B) as explained in Section 2.3.2 has gold standard answers annotated by experts, based on knowledge base triples and plain text paragraphs provided as reference for annotators. The human annotators provided answer spans which are the gold standard answers do not necessarily cover all variants of the answers present in the paragraphs. i.e. these annotations are not an exhaustive list of all answer variants which occur in the paragraphs. They only provide one or two variants.

In this section we discuss and highlight the problem (with examples) and show how a model can perform when the annotation is done while highlighting all the correct variants.

5.1.2 Problems addressed

The evaluation measures computed by BIOASQ for task B phase B are Strict Accuracy, Lenient Accuracy and Mean Reciprocal Rank (MRR). Participating systems are required to provide an answer span (5 answers maximum) for each question. To compute the scores automatically, the exact match of strings between the predictions and the gold standard answers is used to decide if a system answer is correct. Strict accuracy is the rate of Top-1 exact answers. Lenient accuracy is the rate of exact answers in Top-5 predictions. MRR is the mean reciprocal rank computed on the Top-5 system answers.

These measures have been the same since BIOASQ 1, although the first four challenges had triples and concepts along with snippets in the data. In the last three challenges, only relevant snippets for questions are released. Similar evaluations are performed in extractive question answering and reading comprehension tasks like in SQUAD question answering task where Top-1 accuracy and F1 scores are computed by comparing exact matching strings after removing stop words and special characters. One main assumption in reading comprehension task is that the answer strings are substrings of the snippets, which implies that answers have to be extracted from the snippets. In BIOASQ, the answers are curated by human experts by analyzing the triples, concepts, and snippets (or paragraphs). Thus, the BioASQ dataset and evaluation measures are very similar to that of reading comprehension task, but the major difference apart from the dataset size are the answers instances provided as gold standard which does not contain all the occurrences, abbreviations, different forms of answers which are present in the snippets.

As in (Wiese et al., 2017c), we transform BIOASQ Phase B as a reading comprehension task with domain adaptation (we explained this process in detail in Section 4.2.2). Gold standard answer strings and their offsets are automatically searched in the snippets for exact match and treated as answers if only they are found in the snippets, i.e., the answer string must be a substring of the snippet. We term this method as *Distant Supervision* method. By doing so the dataset size is reduced to 65% of Bioasq 5B train set which was suitable for adaptation. Other 35% of the questions did not have matching answers in the snippets, because of different variants of answers in the snippets, missing abbreviations, or irrelevant snippets. We also found that in BIOASQ 6B training dataset for factoid questions, 205 out of 619 questions have false negative answers (33% of the dataset). This kind of corpus annotation may result in some problems:

- Less data for learning.
- The model does not learn to extract all the variants because it does not learn from all the variants.
- Evaluation is done using such gold standard data which will lower the results even though the model is performing well.

This above explained automatic snippet annotation method results in:

- False positive: an answer mentioned in a snippet which does not answer the question. i.e. a snippet alone does not justify the answer to the question.
- False negative: a snippet answers the question but does not have the exact string compared to the gold standard string.

Below are some examples for which the answers returned from a reference system is correct (when evaluated manually) but the automatic evaluation classifies it as incorrect.

Q: Which calcium channels does ethosuximide target?
P: ...neuropathic pain is blocked by ethosuximide, known to block T-type calcium channels,..

Prediction: T-type calcium
Gold standard: T-type calcium channels

Example 1: Missing keywords in predictions (False Negative)

Q: Which disease can be treated with Delamanid?
P: In conclusion, delamanid is a useful addition to the treatment options currently available for patients with MDR-TB.

Prediction: MDR-TB
Gold standard: tuberculosis

Example 2: Abbreviations and their expansions mismatch (False Negative)

Q: Which MAP kinase phosphorylates the transcription factor c-jun?
P: c-Jun NH2-terminal kinases (JNK) play important roles in T helper cell (Th) proliferation, differentiation, and maintenance of Th1/Th2 polarization.

Gold standard: c-Jun NH2-terminal kinase (JNK)

Example 3: Non justifiable paragraph (False Positive)

In example 1, because of a missing word "channels", the predicted answer is marked incorrect. In example 2, MDR-TB stands for *Multi-drug-resistant tuberculosis*, which is from a relevant snippet but since the gold standard has only *tuberculosis*, it is marked incorrect. Contextually both are valid answers. In example 3, the paragraph has no relation with the question except that it contains the answer terms.

To overcome this problem and enrich the answer space correctly, we manually annotated 618 factoid question-answers pairs from the training dataset of 6B task, by annotating the substring of the gold standard answers in the snippets, and adding answers with abbreviations, multi-word answers, synonyms, that are likely correct answers. We explain this in detail in the following section.

5.1.3 Annotations of variants

Manual Annotations

To overcome the issues mentioned above, we manually annotated answer variants in the data. The details of the annotations on the BIOASQ 6B training dataset and the statistics are presented here.

Our annotations include the following type of answers:

- Exact Answer - Exact match with gold standard (GS) answers, which can also be annotated automatically, and different variants of the answers. For example, the annotation of a single GS answer "*Transcription factor EB (TFEB)*" resulted in 3 annotations, "*Transcription factor EB*", "*TFEB*", "*Transcription factor EB (TFEB)*".

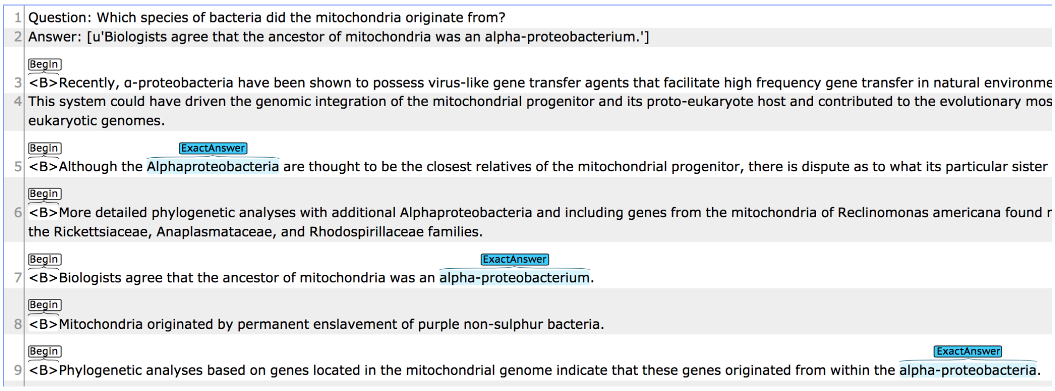
- Lenient Answer - a more general form or a more specific form of an answer. An example is "Telomerase" for "Human telomerase reverse transcriptase".

- Paragraph Answer - The answer matches with gold standard but the snippet alone is not relevant to the question. - this corresponds to the false positive case.

We came across several kinds of snippets. A supporting snippet, or answering snippet, is a snippet that contains the answer and has enough elements for justifying it. It is a correct answer to the question (snippet starting at line 5 in Figure 5.1 for example). A snippet that contains the answer without justification towards the question will not be annotated with the answer as correct and is considered as a non-supporting snippet (snippet starting at line 3 in Figure 5.1). A snippet that does not contain the answer cannot be a supporting snippet, henceforth it is an irrelevant snippet (snippet starting at line 8 in Figure 5.1).

We use Brat[2] annotation tool by (Stenetorp et al., 2012) shown in Fig. 5.1 to perform the manual annotations of the snippets with the answer to the question. The annotations done include the answer string along with their character offsets in the snippet. Answers were annotated by 3 people from computer science background

[2] http://brat.nlplab.org



Question: Which species of bacteria did the mitochondria originate from?
Answer: [u'Biologists agree that the ancestor of mitochondria was an alpha-proteobacterium.']
<B>Recently, α-proteobacteria have been shown to possess virus-like gene transfer agents that facilitate high frequency gene transfer in natural environme
This system could have driven the genomic integration of the mitochondrial progenitor and its proto-eukaryote host and contributed to the evolutionary mos
eukaryotic genomes.
<B>Although the Alphaproteobacteria are thought to be the closest relatives of the mitochondrial progenitor, there is dispute as to what its particular sister
<B>More detailed phylogenetic analyses with additional Alphaproteobacteria and including genes from the mitochondria of Reclinomonas americana found r
the Rickettsiaceae, Anaplasmataceae, and Rhodospirillaceae families.
<B>Biologists agree that the ancestor of mitochondria was an alpha-proteobacterium.
<B>Mitochondria originated by permanent enslavement of purple non-sulphur bacteria.
<B>Phylogenetic analyses based on genes located in the mitochondrial genome indicate that these genes originated from within the alpha-proteobacteria.

Fig. 5.1: Brat annotation tool

and multiple discussions were held to discuss problematic answers which involved looking upon the internet for some medical term meanings.

Annotations were initially done on the BIOASQ 5B training set and the additional questions from 5B test sets whose answers are present in the 6B training set were annotated later on 6B data. So the changes done (if any) on 6B training set for previous year questions from 5B set are not considered.

The annotation files are freely available[3] and can be used by researchers who can obtain the BIOASQ dataset from the official website[4].

	Gold std. annotations		Full annotations	
Count	Avg	Total	Avg	Total
Answers	0.8	500	2.9	1814
Snippets	7.7	3286	8	4965
Questions	-	426	-	618

Tab. 5.1: Annotation statistics

Some statistics of the dataset are listed in Table 5.1 for the automatically annotated answers from gold standard data and the fully annotated data with manual annotations. The annotations are done on 618 BIOASQ 6B training dataset questions. Out of 619 factoid questions, 1 question does not have any snippets. Only 426 questions contain answers from automatic annotation.

"Answers" are the count of answers present in the snippets. *Avg* score represents an average over the total number of questions (i.e. 618). Since in gold standard data, only 426 questions have gold answers in snippets, it is normal for the average to fall below 1. It is clear from the table that the full annotated data contain at least

[3] https://zenodo.org/record/1346193#.W3_WUZMzZQI
[4] http://bioasq.org/

3 times (1814 answers) more the number of candidate answers over the provided gold standard ones (500 answers).

We found that some answers contained the whole snippet as an answer and that 3503 snippets are repeated in the 6B train set. After filtering those repeated snippets we found 3286 different snippets containing exact matching answers extracted automatically from gold standard data and 4965 unique snippets manually annotated with correct answers.

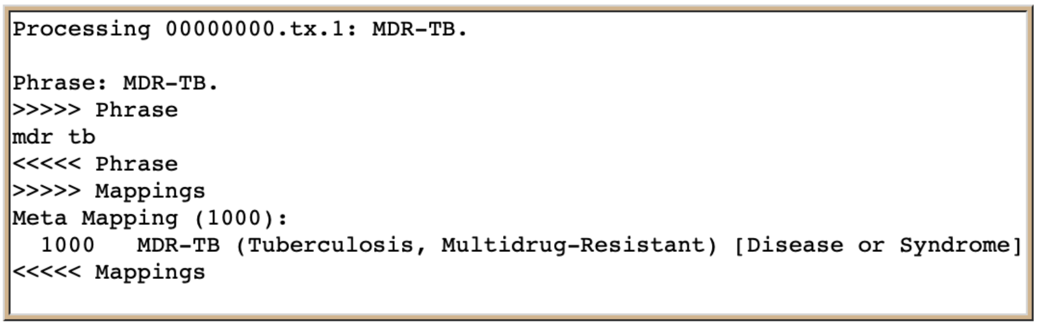
Automatic Annotations

When we realised the problems addressed in section 5.1.2, we did not use an automatic method to annotate answer variants as it was not straightforward to do using string edit distance for abbreviations etc. Instead we proceeded with manual annotations. Once we began annotating it manually, we had some intuitions of using either rule based techniques, noun phrases and text normalizing using some pre-processing techniques.

Manual annotations are expensive in terms of time and money (if experts are paid to annotate). An automatic alternative approach although not very straightforward and accurate as manual ones, is necessary to provide as an alternate solution for future BIOASQ datasets. Hence we propose an automatic way of annotating these answer variants in this section.

Answers often refer to biomedical entities. The answer variants would also refer to the same biomedical entity with different syntactic structures. Since both represent the same biomedical entity, it is easier to detect the entity than the matching strings. We decided to use the UMLS Meta-thesaurus to determine matching entities. *UMLS* - Unified Medical Language System is a Metathesaurus created in 1986, which has become an important and a large resource for biomedical science. It provides over 3,100,000 biomedical concepts imported from nearly 200 vocabularies. Each concept is assigned a Concept Unique Identifier (CUI) that uniquely identifies a single meaning. To consistently categorize these huge number of concepts, 133 Semantic Types are defined in UMLS Metathesaurus. In order to further reduce the complexity of the Metathesaurus, these semantic types are divided into 14 groups, called Semantic Groups[5]. Metamap[6] is a tool that exploits UMLS to annotate free text containing biomedical entities.

[5] https://semanticnetwork.nlm.nih.gov/download/SemGroups.txt
[6] https://metamap.nlm.nih.gov



```
Processing 00000000.tx.1: MDR-TB.

Phrase: MDR-TB.
>>>>> Phrase
mdr tb
<<<<< Phrase
>>>>> Mappings
Meta Mapping (1000):
  1000   MDR-TB (Tuberculosis, Multidrug-Resistant) [Disease or Syndrome]
<<<<< Mappings
```

Fig. 5.2: Output of Metamap tool for a sample paragraph containing the term *MDR-TB*

Figure 5.2 shows a sample output of metamap tool for a paragraph which contains a biomedical term *MDR-TB*. We use PyMetamap code[7] which is a python wrapper program which uses metamap server running in the background and returns metamap annotations. Metamap outputs an identifier for each entity that it recognizes in the text. This identifier is called as a "*CUI*". This identifier is unique for each concept found in UMLS. Metamap also gives the exact paragraph span which triggered this concept while annotating. Using this tool we first annotate the gold standard answer phrases provided by BIOASQ to obtain its "*CUI*". We then annotate the paragraphs with metamap in order to obtain all annotations in them and search for the gold standard answer "*CUI*" in them. The matching "*CUIs*" found this way are the exact same concept in UMLS as the gold standard answer terms, but might be syntactically different such as "Multi-drug-resistant tuberculosis" and "MDR-TB". This allows to determine the UMLS concepts for *Abbreviations*.

There are two cases where automatic annotations fail: 1) *Missing words*: Missing words from answer phrases will not be annotated if it does not belong to the UMLS concepts. 2) Gold standard answers with both *Abbreviations* and the expansion together, as UMLS does not highlight both in the annotations.

To determine if the automatic or the manual annotated ones are better, we experiment by training the model using both data individually and evaluate the results with gold standard data in Section 5.1.4.

5.1.4 Experiments

In this section we report the experiments we performed on the annotated and gold standard data. We follow the process of (Wiese et al., 2017c) and use a machine

[7] https://github.com/AnthonyMRios/pymetamap

reading model developed by (Chen, Fisch, et al., 2017) that is pre-trained on SQUAD dataset (Rajpurkar, J. Zhang, et al., 2016) for open domain questions and fine tuned to biomedical questions. Note that we use the same model for all experiments without changing any model hyperparameters or methods.

To study the impact of training data sets on the evaluations, we train the models using separately the domain adaptation done via *Distant Supervision*, Automatically annotated data and Manually annotated data. We evaluate them individually.

For the detailed explanation of the QA system and domain adaptation to biomedical domain, please refer the section 4.2.2. Several embedding spaces were tested as input vectors (Kamath et al., 2017a) and the best performing ones which were the Glove embeddings trained on common crawl data with 840B tokens, were chosen as input to the system. Unknown words were initialized as zero vectors.

As BIOASQ questions have several answer paragraphs, we treat each question and a paragraph as a training sample which might often result in repeated questions with different paragraphs, i.e. for each training example, there is a question, a unique paragraph and the start and end token string offsets of the answer in the paragraph.

Since there are multiple paragraphs per question, while predicting we consider 5 answers from each paragraph based on the decreasing order of their probabilities, resulting in a set of 5*n answers where n is the number of paragraphs. Top 1 and Top 5 are chosen in this set based on their answer probabilities.

Distant supervised vs Manually annotated answers

In this section we compare the performance of systems trained on answers obtained by the method of *Distant Supervision* and Manually annotated answers.

We perform fine-tuning on two datasets namely

- BIOASQ 5B training set, which contains the 4B training data + the answers of the 4B test data - We term it as 5B.
- BIOASQ 6B training set, which contains the 5B training data + the answers of the 5B test set - We term it as 6B.

We term the distant supervised annotated data as *Gold*, and manually annotated data as *Anno.*, both for train and test splits.

The pre-trained model on open domain QA data is fine-tuned on the above listed Bioasq datasets separately. Evaluation is performed by K-fold cross validation because of the small scale of the data (Table 5.2), and on the official test sets of Bioasq 5B (Table 5.3), which were separated from the training data while fine-tuning.

Train set	5B					6B			
Finetune	Gold			Anno.		Gold		Anno.	
Eval	DeepQA	Gold	Anno.	Gold	Anno.	Gold	Anno.	Gold	Anno.
Strict	-	0.2551	**0.2962**	0.1666	**0.3333**	0.2669	**0.3090**	0.2265	**0.3948**
Lenient	-	0.4156	**0.4444**	0.2991	**0.5843**	0.4417	**0.4724**	0.3511	**0.6197**
MRR	0.2620	0.3138	**0.3425**	0.2148	**0.4322**	0.3334	**0.3718**	0.2728	**0.4765**

Tab. 5.2: K-fold evaluation on different train sets with *Gold* and *Anno* data. DeepQA scores are presented by (Wiese et al., 2017b)

Train set	5B					
Finetune			Gold		Anno.	
Eval	(Wiese et al., 2017c)	Lab Zhu, Fudan Univer	Gold	Anno.	Gold	Anno.
Strict	0.3466	0.3533	0.3533	**0.42**	0.3133	**0.4266**
Lenient	0.5066	0.4533	0.54	**0.64**	0.5	**0.6866**
MRR	-	-	0.4256	**0.5042**	0.3884	**0.5258**

Tab. 5.3: Overall results calculated on official test sets from 5B task. Scores from (Wiese et al., 2017c) and *Lab Zhu, Fudan Univer* are reported in Bioasq 5.

Train set	5B					
Finetune			Gold		Anno.	
Eval	(Wiese et al., 2017c)	Lab Zhu, Fudan Univer	Gold	Anno.	Gold	Anno.
Batch 1	0.5600	0.4200	0.4733	**0.5733**	0.4933	**0.6066**
Batch 2	0.4086	0.4839	0.4274	**0.5510**	0.3387	**0.5215**
Batch 3	0.4308	0.3846	0.4070	**0.4198**	0.3185	**0.3955**
Batch 4	0.3025	0.2601	0.3595	**0.4474**	0.4444	**0.6196**
Batch 5	0.3924	0.4524	0.4271	**0.4771**	0.3452	**0.5023**

Tab. 5.4: MRR results calculated batchwise on 5B official test sets.

The explanation of scores reported in table 5.2 and 5.3 along with the corresponding experiments on the datasets listed above, is as follows. On the data of *Train set* mentioned in the first row, we fine-tune it with *Finetune* data on the second row - which is *Gold* or *Anno.* version of the answers.

The official evaluation measures[8] using *Gold* or *Anno.* version of the test answers are highlighted in the third row. The strict and lenient accuracies along with the MRR are reported.

[8] https://github.com/BioASQ/Evaluation-Measures

Gold version of 5B data contains 313 questions and *Gold* version of 6B data contains 428 questions. We consider the remaining questions with no matching answers as incorrectly answered, hence evaluating over all the questions of the datasets (5B - 486 questions, 6B - 618 questions). Annotated 5B data contains 483 questions and 6B data contains 618 questions.

Overall results of 5B test sets presented in Table 5.3 are evaluated on 150 questions from the test sets of 5B challenge whose gold standard answers are present in 6B challenge train set.

To compare our scores with the ones reported in (Wiese et al., 2017b) and also since the size of the dataset is small, we perform K-Fold (5) evaluations which are reported in Table 5.2. To compare with previously reported official test scores in Bioasq 5, we train on 5B training set and test on 5B test sets which are reported in Table 5.3.

The results shown in the tables 5.2, 5.3 and 5.4 highlights the improvements using manually annotated data over the distance supervised annotated data on the QA performance as well as the evaluations with *Gold* and *Anno.* versions of answers.

In Table 5.2, training on *Gold* and evaluating on *Gold* are the baseline scores. *DeepQA* MRR score is the K-fold evaluation score of MRR reported on 5B train set by (Wiese et al., 2017b). Comparing the *DeepQA* MRR score with the *Gold* and *Anno.* 5B versions, there is an improvement of at least 17% (*Anno.* training and *Anno.* evaluation) to 8% (*Gold* training and *Anno.* evaluation).

In terms of accuracy, training the model on *Anno.* version and evaluating on *Anno.* version of answers fetch best results by 3.68% and 8.58% on Strict accuracy, 14% and 14.73% on Lenient accuracy in 5B and 6B respectively.

Training on *Anno.* and evaluating on *Gold* has low scores in almost all experiments because of the model which learns on different forms of answers, therefore predicts different forms of answers which are not present in the *Gold* version.

In Table 5.3, because of a low number of questions in the official test sets ranging from 25 to 35 questions for each batch, the scores are computed over all 5B batch test sets (5B test sets - 150 questions). The scores by (Wiese et al., 2017c) and *Lab Zhu, Fudan Univer* are the best official results in Bioasq 5. We calculated strict and lenient accuracy as mentioned above and our scores are better than both best official results by 6.67% for strict accuracy and 13.34% lenient accuracy on *Gold* version training, 7.33% for strict accuracy and 18% lenient accuracy on *Anno.* version training.

In Table 5.4, MRR scores are reported separately for each batch. MRR scores in general have the best scores compared to both (Wiese et al., 2017c) and *Lab Zhu, Fudan Univer* by training on *Anno.* and evaluating on *Anno.* versions.

Automatically annotated vs Manually annotated answers

To determine the usefulness of automatically annotated answers done using Metamap with the manual annotated ones, we use the same methods to train as above done for manually annotated answers, but train with automatically annotated answers which are obtained as explained in section 5.1.3 of this chapter. To avoid confusion with distant supervised data and automatically annotated data by metamap, we present the results separately.

Evaluation	-	BIOASQ 5B		BIOASQ 6B	
-	Measures	Manual	Auto.	Manual	Auto.
Gold Data	Strict	25.62	**27.28**	**38.68**	38.12
	Lenient	**43.82**	39.99	59.00	**60.64**
	MRR	31.95	**32.32**	46.62	**46.91**
Automatically Annotated Data	Strict	**45.83**	37.81	47.59	**48.59**
	Lenient	**70.23**	62.34	**75.68**	74.70
	MRR	**53.90**	46.55	58.39	**58.53**

Tab. 5.5: Experiments with manual and automatically annotated dataset. Evaluation done on both BIOASQ gold standard data and annotated datasets.

In this set of experiments, we try to determine which method of annotation is better and if automatic annotation is comparable with manual annotations. If automatic annotations provide better or similar performance, this avoids the need of expensive manual annotations done by experts.

Table 5.5 shows the results of the models trained on automatically annotated data and manually annotated data. *Gold Data* evaluations are done on gold standard answers given by BIOASQ challenge and *Annotated Data* evaluations are done on respective annotated data. The results show that both automatic and manual annotations are comparable on 6B set. *Lenient accuracy* (Top 5) is better with manual annotated data than the automatically annotated data (3 out of 4 experiments). And for *Strict accuracy* (Top-1) both perform very similar except BIOASQ 5B data where manual data is better when evaluating on *Annotated dataset*.

Although the data used in the Table 5.5 for *Manual annotated dataset* and in the previous section of experiments in the Table 5.2 for *Annotated dataset* is exactly the same, the results on *Gold Data* evaluation differ largely between these two tables on *Manual annotated dataset* training. The scores in Table 5.5 is much higher than Table 5.2 because of the difference in the prediction module. In the method presented in

Table 5.2, we predict 5 answers per paragraph based on decreasing order of answer probabilities and choose Top 1 and Top 5 based on answer probabilities of all the resulting candidates. In the method presented in Table 5.5, if we have 1 paragraph per question then we take 5 predictions from the same paragraph as Top 5 answers and rank them based on the decreasing score of answer probabilities. If we have more than 1 paragraphs per question, we take at least 1 prediction from a paragraph which has maximum probability until we have at least 5 candidates and choose Top 1 and Top 5 based on answer probabilities of all the resulting candidates. The order of choosing the paragraphs for answer extraction is the same as provided by BIOASQ gold standard data.

These small changes in the prediction module has a large impact on the result mainly because the second method considers answers from paragraphs whose top answers might not have the highest probability compared to another paragraph top answer.

The experiments and results show that the automatically annotated dataset using Metamap performs similar to the manually annotated datasets henceforth this method is better, easier and cheaper to adapt future datasets to cover more answer variants for biomedical domain. The study reported above shows the influence of enriching the training data by manually and automatically annotating variants of gold standard answers, on the evaluation performance. We used UMLS Meta-thesaurus as a source for Metamap tool to detect and annotate biomedical vocabulary. We show the impact of the enriched data on *Reading Comprehension task* by experimenting on two training datasets. Our method outperforms some of the best-performing systems from BIOASQ without changing the model.

In the following section, we report our studies on using semantic features such as Expected Answer Types both in open domain and biomedical domain on improving the QA model performance in different tasks.

5.2 Expected Answer Types (EAT) and their importance in Question Answering models

5.2.1 Introduction

In traditional QA systems (non deep learning approaches) on text, one of the main criteria for selecting an answer is based on recognizing the Lexical Answer Type (LAT) and the Expected Answer Type (EAT) in order to do a matching with candidate answers.

The Expected Answer Type (EAT) is a type which determines the type of answer for a question. This type is inferred from the question. In open domain, named entity types can be used as EAT. Table 5.6 shows the taxonomy we follow to obtain EAT from the named entity types. We use Spacy tool[9] to determine named entities and map them to the Expected Answer Type.

Spacy annotated named entity type	EAT
PERSON, ORG, NORP	HUM
LOC, GPE	LOC
PRODUCT, EVENT, LANGUAGE, WORK_OF_ART, LAW, FAC	ENTY
DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL	NUM

Tab. 5.6: Spacy tool uses this named entity annotation scheme following OntoNotes 5 corpus, which is mapped with EAT types

The Lexical Answer Type (LAT) is the word or words in a question which determines the type of expected answers. LAT and EAT are often used in the same context but are very different from each other. LAT corresponds to the word in the question phrase which is used to infer an expected type, whereas an EAT is the inferred type and not a word in the question. Some examples of questions and answers with their lexical answer types and expected answer types are given below:

Question: What was the name of the food chain owned by Gus Fring?
Answer: Los Pollos Hermanos
Expected Answer Type (EAT): ORG.
Lexical Answer Type (LAT): food chain.

Question: What was the food joint famous for?
Answer: Fried chicken
Expected Answer Type (EAT): NO_EAT
Lexical Answer Type (LAT): What

Question: Who played the role of Walter White in the series Breaking Bad?
Answer: Bryan Cranston
Expected Answer Type (EAT): HUM.
Lexical Answer Type (LAT): Who.

[9] https://spacy.io/

Question: Where was the TV show Breaking Bad primarily filmed at?
Answer: Albuquerque
Expected Answer Type (EAT): LOC.
Lexical Answer Type (LAT): Where.

Question: Which president of the USA did not sign the Paris climate agreement?
Answer: Donald Trump
Expected Answer Type (EAT): PER
Lexical Answer Type (LAT): President

In biomedical QA, the answers do not belong to the open domain named entity types as shown in Table 5.6. Biomedical domain Expected Answer Types (EATs) are specific to the domain.

In the case of biomedical domain for entities, the UMLS gives references of semantic types and groups which give more granular information about the entity type which it belongs to. Therefore instead of named entity types from tools like Spacy, we use semantic types and groups from UMLS[10]. There are 133 semantic types and 14 semantic groups. Metamap tool[11] is used to detect these for biomedical entities.

Some examples of biomedical questions and answers with their lexical answer types and expected answer types are given below:
Question: What disease in Loxapine prominently used for?
Answer: Schizophrenia
Expected Answer Type: Disease
Semantic Group: DISO Disease or Syndrome.
Lexical Answer Type: disease.

Question: Which drugs are utilized to treat amiodarone-induced thyroitoxicosis?
Answer: Antithyroid drugs
Expected Answer Type: Drug
Semantic Group: CHEM Chemicals & Drugs
Lexical Answer Type: drugs.

Question: What causes the majority of lung cancers?
Answer: Smoking

[10] https://mmtx.nlm.nih.gov/SemanticTypesAndGroups.shtml
[11] https://metamap.nlm.nih.gov/

Expected Answer Type: NO_EAT
Semantic Group: None
Lexical Answer Type: causes

Traditional QA systems on text are made of several pipeline modules: question analysis, passage selection, answer selection. Question analysis allows to extract features that are used for selecting passages and extracting the answer. Apart from the plain textual words, these features can be different from a system to another, but they all make use of the Expected Answer Type (EAT) (Kolomiyets and Moens, 2011). The definition of Expected Answer Type (EAT) and some examples are presented above. Best methods for verifying if a candidate answer matches the EAT involves feature based supervised learning based on the use of different resources, as co-occurrences and presence in structured resources (Grappy and Grau, 2010; Grappy, Grau, et al., 2011; Chu-Carroll et al., 2012). In medical domain, this verification was made using UMLS (Zi Yang et al., 2016; Abacha and Zweigenbaum, 2015).

Recent QA approaches are based on deep neural network architectures, mainly in the open domain. The authors of (Weissenborn et al., 2017) for their RNN based model introduce a supplementary feature that is the word embeddings of LAT. It is defined as the question word or the words around the question word (two different features) whose embeddings are averaged and appended as a feature vector to the model. However they did not report results that allows to evaluate the impact of EAT on the overall performance of the model.

For the studies presented in this chapter, some of the resources used are listed below:

UMLS is used extensively in almost all our works involving biomedical domain data. For example, a concept such as *Transcription factor EB (TFEB)* has a CUI C1420699 in the UMLS and belongs to the *[Gene or Genome]* semantic type. If a textual paragraph contains a term "TFEB" or just "Transcription factor EB" without the abbreviation, it belongs to the same concept mentioned above and UMLS tool Metamap can be used to map these text spans to the concept to determine its features and relations with other concepts.

Semantic types and semantic groups have been used in various biomedical information systems, including categorizing clinical research eligibility criteria (Luo et al., 2011), learning biomedical ontology (Petrova et al., 2015), and representing clinical questions for medical QA (Kobayashi and Shyu, 2006).

Pubmed articles are used to create word embeddings for biomedical domain words which are used for tasks such as Named Entity Recognition (NER) by (Habibi et al., 2017), Question Answering (QA) by (Tsatsaronis et al., 2015; Wiese et al., 2017c).

In the following sections we present a study to understand the usefulness of *Expected Answer Types (EAT)* in Question Answering systems in biomedical domain and open domain.

5.2.2 Verification of the Expected Answer Types in Biomedical Domain

In the first study, we begin with biomedical domain data. To understand the importance of Expected Answer Types in biomedical QA dataset, one set of experiment involves analysing the output of our QA system for determining how many system responses correspond to the *Expected Answer Type* from the question. For this purpose we use the *Reading Comprehension task* model *DRQA* by (Chen, Fisch, et al., 2017) which is explained in Section 4.1.1.

The model does not make use of any LAT or EAT information or any medical domain related resources. The model only relies on input data tagged with named entities and encoded with word embeddings. Thus the goal is to study if it can be interesting to add information regarding the *Lexical Answer Type* or *Expected Answer Type* into the model explicitly. Alternatively we can also determine if the model already captures the *Answer Type* information by analysing the predictions.

For the analysis, we take the gold standard answers provided by the BIOASQ and use Metamap tool[12] to annotate the answers to determine their semantic types[13] from UMLS. These semantic types are our gold standard answer Expected Answer Types (EAT). We determine the percentage of EAT matches provided by the gold standard answers and the QA system's response to analyze two aspects:

- If the Expected Answer Type inferred from the question by using the Lexical Answer Type (LAT), matches gold standard answer types.
- If the Expected Answer Type inferred from the question is already captured by the QA system's response even when the answer is wrong.

[12] https://metamap.nlm.nih.gov
[13] https://mmtx.nlm.nih.gov/SemanticTypesAndGroups.shtml

As we explain in the previous section, an Expected Answer Type (EAT) is the type of the answer, whereas a Lexical Answer Type (LAT) is a word or words in the question which is used to infer the EAT.

For the biomedical domain, (Neves and Kraus, 2016) released a corpus named *BiomedLat* which consists of LAT and EAT annotations for BIOASQ questions[14] which were manually annotated with LAT words into them and their semantic group from UMLS which are the EAT.

For our study, we consider different representations for the EAT from the questions:

- The Semantic Group of LAT. - This refers to the LAT from the question, whose semantic groups from the UMLS semantic network[15] are used - **SGEAT - Semantic Group Expected Answer Type**.

- a word embedding for LAT (**WEEAT - Word Embedding Expected Answer Type**) - word embeddings for LAT words obtained using a pre-trained word embedding space. We use this to measure cosine distance between answer word embeddings and the LAT embeddings (WEEAT). For computing WEEAT when the LAT is made of several words, we compute the average of each word embedding of the LAT. When a word has no embedding, we set its vector to 0. We use Word2Vec skipgram model with 300 dimensions from (Tomas Mikolov, Sutskever, et al., 2013) for computing word embeddings on the biomedical texts of BioAsq 5A task data which consist of 12.8 Million PUBMED articles.

To determine if the Expected Answer Type can be useful for selecting an answer, we study if the EAT given in the *BiomedLat* corpus by (Neves and Kraus, 2016) (the gold standard EAT for questions (**GoldEAT**)) matches with the BIOASQ answer inferred semantic types (the answers in the gold standard BIOASQ data (**GoldAns**)) and with the answer semantic types for predicted answers by our QA system (**PredAns**). Semantic types from the answers (gold or predicted) are inferred using the Metamap tool which annotates the answers with a semantic type.

An example to explain the above terms:
Question: What disease in Loxapine prominently used for?
Gold Standard Expected Answer Type (**GoldEAT**) : Disease
Gold Standard Lexical Answer Type: disease.
Gold Standard Semantic Group: DISO Disease or Syndrome.

[14] https://github.com/mariananeves/BioMedLAT

[15] https://metamap.nlm.nih.gov/SemanticTypesAndGroups.shtml

Answer (BIOASQ given): Schizophrenia
Answer Expected Answer Type (**GoldAns**): Disease
Predicted answer (QA model output): Clozapine
Predicted answer inferred Expected Answer Type (**PredAns**): Drug

In the above example, GoldEAT is *Disease*. GoldAns is *Disease*. Therefore the gold standard answer given by BIOASQ matches the GoldEAT which is an expected result.

PredAns is *Drug* which is inferred from the predicted answer *Clozapine* which is a wrong answer for the question, therefore the answer types do not match.

Experiments and Results

For the experiments, we consider only the factoid questions from BiomedLat corpus. We split the dataset into train and test sets (80% train and 20% test). The statistics reported in the Figure 5.3 are for the factoid question test set.

We compute cosine similarities between LAT word embeddings in questions and three different answer word embeddings which are detailed below:

- GoldStandard-maxCosine (crossed points): Answer words are Gold standard data annotated with all answer representations that have the maximal cosine similarity with WEEAT.

- DRQA-cosine-top1 (triangular points): Answer words are Top-1 answers from DRQA output. The similarities of correct (resp. false) answers are plotted above (resp. below) the X-axis.

- DRQA-maxCosine (round points): Answer words are from top 5 answers of DRQA output that have the maximal cosine similarity with WEEAT. The similarities of correct (resp. false) answers are plotted above (resp. below) the X-axis.

From Figure 5.3, we can see that gold standard answers (GoldStandard-maxCosine) show a significant correlation with LAT in terms of word embeddings, although there are 6 questions whose LAT have 0 similarity with WEEAT caused by missing word embeddings for the medical domain vocabulary. Another clear observation is that many of top-1 wrong answers from DRQA system have low similarities (less than

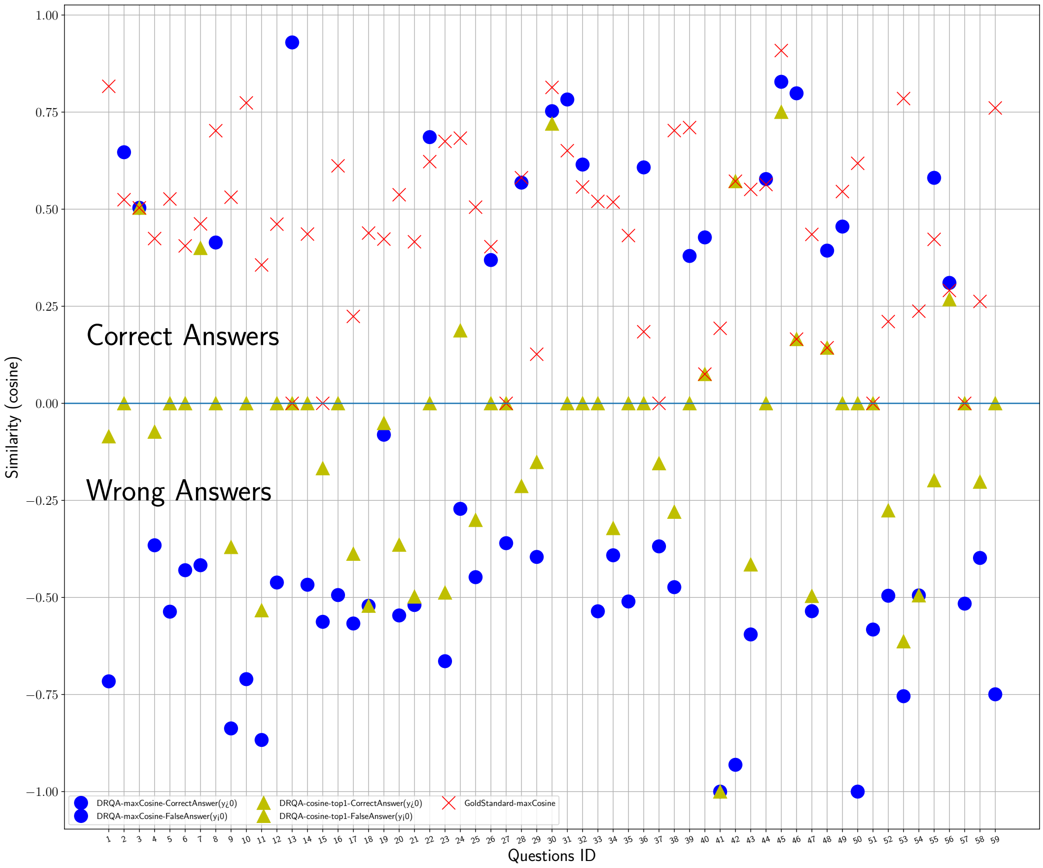


Fig. 5.3: The distribution of answers in three different answering settings for 59 questions: the red crossed points are for gold-standard answers that have the maximal similarity with the question LAT word embedding; So all the crossed points are correct answers. The blue round points above the X-axis are correct answers returned by DRQA system with maximal cosine similarity with WEEAT; The round points under the X-axis are false answers found by DRQA system with maximal cosine similarity. The absolute value is the similarity. The green triangles stand for the top-1 results of DRQA system, where the upper parts are correct answers and the low parts are wrong answers.

0.25), which indicates that we could remove some wrong answers according to this criterion.

Moreover, Figure 5.3 shows that there are around 50% top-1 answers having zero similarity with question LAT. This could be caused by the out-of-vocabulary problem of word embeddings such as short answers with specific words that have never appeared in the training corpus.

For the round points below the X-axis, they also present an important similarity (around 0.5) correlation with WEEAT, which means that by simply selecting the answer with highest similarity as the best answer is not an effective strategy. Indeed, when we used this re-ranking strategy to select one answer from DRQA candidate answers, the strict accuracy with respect to the annotated gold standard decreased from 38% to 33%. Again, the missing word embedding for correct answers has a strong impact on this results.

The observations above show that a fine-grained study of word embeddings is important for biomedical QA systems and missing vocabulary for biomedical terms might cause performance degradations.

Tab. 5.7: SGEAT (Semantic Group Expected Answer Type) associated to answers

Dataset	Answer count
Gold standard data	40/59
DRQA correct top-1 output	18/23
DRQA wrong top-1 output	16/36

To determine the importance of semantic types inferred from the lexical answer types from question (SGEAT) in answer words, we studied if the SGEAT types are present in the answers. We report this on three datasets, one being the Gold standard answers in BIOASQ and other two being the correct and wrong predicted answers of DRQA system (top-1).

Table 5.7 shows the count of matches of SGEAT and semantic types from answers. It is clear that many correct answers (gold standard - 40/59) have a matching SGEAT. For DRQA outputs, we compute how many correct and wrong top-1 answers has a matching SGEAT. From the reported findings, there are more correctly answered DRQA outputs (18/23) with matching SGEAT than the wrong ones (16/36) which signifies that the model already captures this information correctly in most of the cases. But 44% (16/36 questions) of wrongly answered questions have the matching Expected Answer Types, which means that this feature is useful to improve wrongly answered questions.

We studied different representations of the LAT words and EAT, based on structured taxonomy or word embeddings, and showed a correlation with the correct answers. When comparing with the answers provided by our QA model, we observe that the wrong answers might be rejected by adding a criterion when the answer types do not match.

5.2.3 Verification of the Expected Answer Types in Open Domain

In our study of the open domain, we analyze the SQUAD dataset by (Rajpurkar, J. Zhang, et al., 2016) for the presence of *Expected Answer Types*, which is based on Wikipedia data. For analyzing this data, we need LAT and EAT information on the SQUAD dataset. The authors of (Madabushi and M. Lee, 2016) built a rule based model which is highly accurate (97.2% in their evaluations) in determining the question classes for input questions in a taxonomy defined in their work. We obtain these annotations on SQUAD dataset questions upon request which we use for the study of EAT for open domain.

We have created a simple taxonomy by using the higher level taxonomy from their work to a version which is suitable for our work as shown in the Table 5.8. In open domain data setting, the named entity types are mainly used to define the Expected Answer Types (EAT). A taxonomy must be defined for mapping named entity types to Expected Answer Types (EAT) of the questions. Table 5.8 shows the taxonomy we used.

EAT	Spacy annotated tag
HUM	PERSON, ORG, NORP
LOC	LOC, GPE, NORP
ENTY	PRODUCT, EVENT, LANGUAGE, WORK_OF_ART, LAW, FAC, NORP
NUM	DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL
ABBR	-
DESC	-

Tab. 5.8: Named entity annotation scheme following OntoNotes 5 corpus mapped with EAT.

The annotations given by (Madabushi and M. Lee, 2016) contain high level EAT mentioned in the table 5.8 on the left. The right side values "spacy annotated tags" are the tags which are annotated by named entity recognition tool *Spacy*[16] which the answer words belong to. The mapping determines the which named entity type belongs to which Expected Answer Type (EAT). Since *ABBR - Abbreviations, DESC- Description* cannot be inferred from the named entities as they refer to textual phrases and not exclusively entities, we exclude these from the analysis. These types are present in Figures 5.4 and 5.5 because the gold standard data were not inferred using the named entity types.

The annotations on SQUAD dataset by (Madabushi and M. Lee, 2016) contained the following numbers of EAT as shown in the figures 5.4 and 5.5 across 6 different categories of EAT.

[16] https://spacy.io/

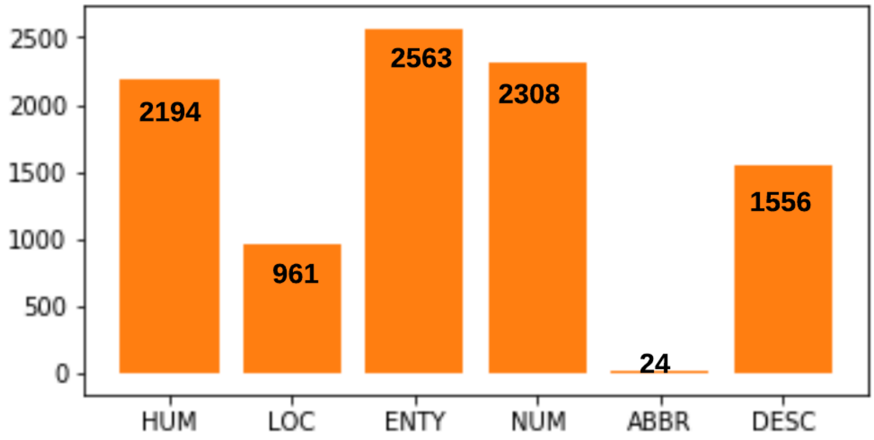


Fig. 5.4: EAT statistics on SQUAD dev set - contains 8026 annotations for 10,570 questions.

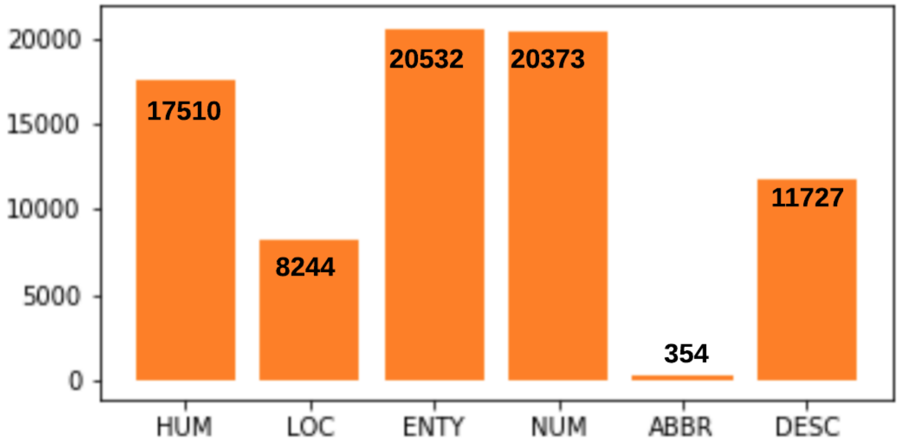


Fig. 5.5: EAT statistics on SQUAD train set - contains 66,659 annotations for 87,599 questions.

We annotate answer spans using the spacy named entity recognition tool and map it with the EAT taxonomy as highlighted above to check how many of them match correctly. For the dev set: There are 8026 questions with EAT annotations (subtracting 24 ABBR and 1556 DESC tagged questions from 9606 total questions). 4029 questions out of 8026 match according to the EAT annotations, i.e. 50.19%. For the train set: There are 66,659 questions with EAT annotations (subtracting 354 ABBR and 11727 DESC tagged questions from 78,740 total questions). 31,924 questions out of 65,419 match according to the EAT annotations. i.e. 48.79%.

The above analysis shows that EAT match happens for 50.19% of questions on dev set and 48.79% of questions on train set. The non matching questions might be because of several reasons:

- Correct Expected Answer Type not being recognized by the rule based system annotations of (Madabushi and M. Lee, 2016).
- Non recognition or wrong recognition of certain entities and their types in the answers from the named entity recognition tool Spacy.
- Wrong taxonomy mapping for certain types. For ex: NORP named entity refers to HUM, LOC and ENTY EAT types.

From the scores reported above, we observe that it is difficult to use EAT as a strong feature for selecting an answer from the paragraph.

Experiment using QA model

In order to understand how many wrongly answered questions correspond to their Expected Answer Type (EAT), we experiment using QA model predictions. We only consider the questions which were wrongly answered to study if the predicted answer type corresponds to the correct expected answer type from the question.

For this set of experiments we use *DRQA* model on *Reading Comprehension task*. We train on the official SQUAD train set and predict on the official SQUAD dev set. We analyze these wrong answers to check if they match the Expected Answer Types from the questions.

The official dev set has 10,570 questions and 3,242 questions (30.67%) are wrongly answered by *DRQA* model. Out of the 3242 questions only 721 questions (22.23%) has an EAT from the annotated set provided by (Madabushi and M. Lee, 2016) which matched the named entity inferred answer type.

This signifies that the QA model predicted only a few number of answers (22.23%) in the wrongly answered set which matched the corrected EAT but failed to capture the right entity as the answer. This shows that there is a scope for highlighting entity types as 77.77% of wrongly answered questions (2,521 questions) did not have a matching EAT. Since 22.23% did not have correct answers in spite of their matching answer types, shows that the entities were not correctly recognised.

In the following section we address these issues by highlighting EAT in the paragraphs and using EAT in the QA models.

5.3 Using Expected Answer Types and Embeddings for the Answer Sentence Selection task

Expected Answer Types (EAT) is one of the vital information which is important for question answering systems to detect which type of answers do the questions require and filter out the less important answer candidates. We hypothesize that this information can be explicitly highlighted and used in the existing models to improve performance.

In the *Answer Sentence Selection* task, a question and a set of sentences are provided and the goal is to find which sentence is the correct answer to the question. There can be multiple correct answer sentences per question. Our intuition is that when the type of an entity (among several other entities in the sentence) present in the sentence corresponds to the Expected Answer Type (EAT) from the question, this sentence is more likely to be the correct answer sentence than a sentence that does not contain such an entity type. Our goal is to better rank these kinds of sentences. We proceed by highlighting, in the paragraphs, entities that correspond to the EAT. We explain the process in detail in the following sections.

Our contributions in this regards are as follows. We introduce two different ways of using Expected Answer Type *EAT*. We use a simple model of a recurrent neural network which uses a pre-attention mechanism. We experiment with several datasets along with TrecQA to determine if this would work better for a wider range of large scale datasets. To annotate other datasets with EAT information, we propose a multiclass classifier model which is trained on a dataset built by using an existing rule-based system which predicts EAT for questions.

5.3.1 Highlighting Single Entity and Multiple Entity Types

An answer sentence contains several named entities and some of them correspond to the Expected Answer Type (EAT). Highlighting these entities using their type might help in improving the QA system performance. The authors of (Tayyar Madabushi et al., 2018) propose a method for replacing words by special token embeddings for highlighting entities that catch the EAT entity in sentences. In our work, this method is referred as “EAT (single type)" in the following experiments. The entities belong to (HUM, LOC, ABBR, DESC, NUM or ENTY). HUM refers to a description, group, individual, title. LOC refers to city, country, mountain, state. ABBR refers to abbreviation, expansion. DESC refers to a definition, description, manner, reason. NUM refers to numerical values such as code, count, date, distance, money, order etc. ENTY refers to a numerous entity types such as animal, body, color, creation, currency, disease etc. More details regarding the taxonomy can be found in (Madabushi and M. Lee, 2016).

The entities, irrespective of which class they belong to, are treated similarly by replacing them by two special tokens *entity_left* for entity occurrences and *max_entity_left* for maximum occurring entity that corresponds to an entity that is at least twice the number of occurrences when compared to the second maximum occurring entity. Entity types are recognized using a named entity recognition tool. When an entity type in a sentence matches the EAT from the question, *entity_left* token is used to replace the entity mentions in the sentences; same applies for the maximum occurring entity token *max_entity_left* as well.

Our proposition is to replace an entity according to the type it belongs to, instead of replacing all kinds of entity by just one word i.e. *entity_left*. We do it based on the different types of EAT it belongs to based on the taxonomy used in the original work. The intuition behind this method is that the model would learn to better map the relations between question words and specific entity type tokens when used in a model with attention mechanisms, rather than learning the relation between question words and the same generic entity type token for all entities. This way, we can learn a different behaviour with an entity about location and with an entity about a person for example.

5.3.2 Answer Sentence Selection Model - RNN-Similarity

The answer sentence selection task is a question answering task which is also referred sometimes as *Sentence Reranking* or *Sentence Ranking* task. It involves ranking a set of sentences $S = \{S_1, ..., S_m\}$ for a given question Q, so the correct answer sentences are ranked higher. Sentence set S can contain the mixture of both negative and positive sentences relevant to the question, often more than one positive sentence.

We model this task as a pairwise similarity scoring task. For each sentence related to a question, we compute a similarity score against the question sentence and answer sentence. i.e., ($Q_i - S_{i,j}$, $Q_i - S_{i,j+1}$, $Q_i - S_{i,j+2}$, ...$Q_i - S_{i,j+n}$).

Recurrent neural networks such as LSTMs and GRUs are widely used in several NLP tasks like machine translation, sequence tagging, and question answering tasks such as reading comprehension and answer sentence selection. We propose a simple model with recurrent neural networks and an attention mechanism to capture sequential semantic information of words in both questions and sentences and predict similarity scores between them. We refer to this model further in this article as *RNN-Similarity* model whose code is available online[17]. Figure 5.6 shows the architecture of the model.

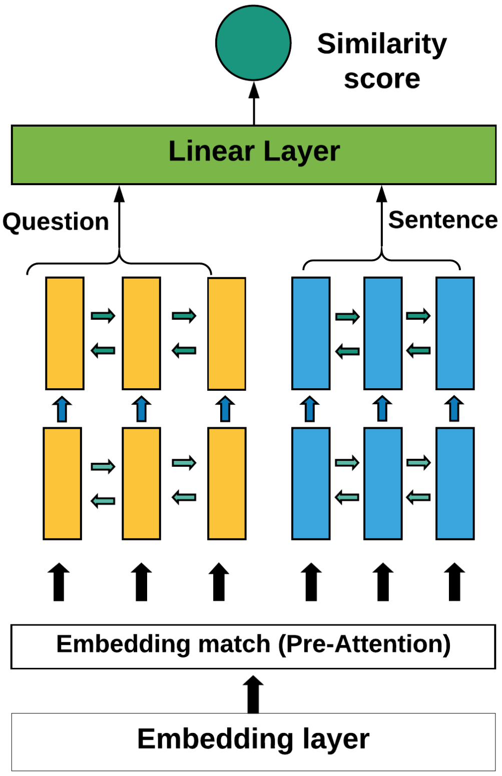


Fig. 5.6: Proposed RNN-Similarity model

Question words $Q = \{q_1, ..., q_m\}$ and Sentence words $S = \{s_1, ..., s_n\}$ are sequences which are encoded using an embedding layer of dimension D.

$$E(Q) = \{E(q_1), .., E(q_m)\} \tag{5.1}$$

$$E(S) = \{E(s_1), .., E(s_n)\} \tag{5.2}$$

A pre-attention mechanism captures the similarity between sentence words and questions words in the same layer. For this purpose, a feature $\mathcal{F}align$ shown in Equation 5.3 is added as a feature to the LSTM layer.

[17] https://github.com/rsanjaykamath/RNN-Similarity

$$\mathcal{F}align(p_i) = \Sigma_j a_{i,j} E(q_j) \tag{5.3}$$

Where $a_{i,j}$ is,

$$a_{i,j} = \frac{exp\,(\alpha(E(s_i)) \cdot \alpha(E(q_j))}{\Sigma_{j'}\; exp(\alpha(E(s_i)) \cdot \alpha(E(q_{j'}))} \tag{5.4}$$

which computes the dot products between nonlinear mappings of word embeddings of question and sentence.

The above process is similar to the *DRQA* model by (Chen, Fisch, et al., 2017) and *PSPR* model by (Y. Lin et al., 2018) for both *Paragraph Selector* and *Paragraph Reader* who use LSTMs to encode Question and Paragraph words along with the pre-attention mechanism as shown in Equation 5.3 . We use 3-layer Bidirectional LSTM layers for both question and sentence encodings.

$$\{E(q_1), .., E(q_n)\} = \text{Bi-LSTM}(\{\tilde{E}(q_1), .., \tilde{E}(q_n\}) \tag{5.5}$$

$$\{E(s_1), .., E(s_n)\} = \text{Bi-LSTM}(\{\tilde{E}(s_1), .., \tilde{E}(s_n\}) \tag{5.6}$$

The LSTM output states are further connected to a linear layer and a sigmoid non-linear activation function is applied on the output of the linear layer which outputs the score ranging between 0-1, which signifies the similarity between the question and the answer sentence.

We implement the RNN-Similarity model in Pytorch, and we use MSELoss (Mean Squared Error loss) to minimize the error of predictions for relevance scores. The code for the model along with default hyperparameters is publicly available on Github [18]. The QA task is presented in section 5.3.2 in detail. The model's hyper-parameters remain the same for all input settings. Only the modified input text for question and paragraph as shown in section 5.3.1 is the change which shows differences in performance. We use adamax optimizer and keep the missing words as zero vectors. We create a random word embedding ranging between (-0.5 - 0.5) with dimension D for each of the EAT words and encode the word with this embedding when it appears in all our experiments.

-	Method	Question	Sentence
1	Original text	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher'	in 'The Iron Lady,' *Young* traces ...... the greatest woman political leader since *Catherine the Great*.
2	Replacement - (Tayyar Madabushi et al., 2018) (EAT Single type)	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher' *max_entity_left entity_left*	in 'The Iron Lady,' *max_entity_left* traces ...... the greatest woman political leader since *entity_left*.
3	EAT (Different types)	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher' *max_entity_left entity_hum*	in 'The Iron Lady,' *max_entity_left* traces ...... the greatest woman political leader since *entity_hum*.
4	EAT (MAX + Different types)	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher' *max_entity_hum entity_hum*	in 'The Iron Lady,' *max_entity_hum* traces ...... the greatest woman political leader since *entity_hum*.

Tab. 5.9: Three methods for replacing entities along with an example from TrecQA dataset

5.3.3 Highlighting Process

The example in Table 5.9 shows the highlighting process. The row 1 is the original text data from the dataset. The row 2 is the modification done by (Tayyar Madabushi et al., 2018) to highlight maximum occurring entity and the EAT matching entity. The row 3 and row 4 are our modifications. The row 3 refers to an example that has an EAT as "HUM" from the taxonomy, so we replace it as *entity_hum*. We do the same for other expected answer types such as *entity_loc* for " LOC" type, *entity_enty* for " ENTY" type, *entity_num* for "NUM" type, *entity_desc* for "DESC" type, *entity_abbr* for "ABBR" type. We replace the entity mentions in the text whose types are matching the EAT from questions same as the authors of (Tayyar Madabushi et al., 2018), but we use separate entity highlighting according to their type, instead of using *entity_left* for all entities.

We also experiment with a variant where the *max_entity_left* is replaced with the entity type along with other entities as done in row 4 of Table 5.9. If the maximum

[18] https://github.com/rsanjaykamath/RNN-Similarity

entity is of type “HUM", we replace it as *max_entity_hum*. This method is referred to as “EAT (MAX + different types)" in the following experiments.

5.3.4 Prediction of the EAT

In order to experiment on the datasets without EAT annotations, we had to develop our own annotation tools to annotate any dataset questions with EAT entities.

Since SQUAD-EAT (see section 5.3.5) is the result of a rule-based method with a high accuracy score (97.2% as reported in (Madabushi and M. Lee, 2016)), we use it to train a multiclass classifier based on a CNN model for text classification[19] by (Kim, 2014), by modifying the outputs into a multi-class setting. We use the original CNN model by (Kim, 2014) built for binary classification of sentiments because it captures better semantic information from text than traditional ML models as pointed in their work which also uses word embeddings from Word2vec (Tomas Mikolov, Sutskever, et al., 2013) as input for their task. We further refer to our modified model as EAT Classifier. We use 300 dimensions GloVe embeddings by (Pennington et al., 2014). The output classes of the classifier refer to a type based on the taxonomy such as *ABBR, DESC, ENTY, HUM, LOC, NUM* and a "NO_EAT" class to signify an EAT which is not in the above list of classes.

We train the multi-class classifier model using the SQUAD-EAT dataset which gets an accuracy score of 95.17% on the SQUAD-EAT dev in our experiment, according to the annotation done by (Tayyar Madabushi et al., 2018) as reference. We release the code for the EAT Classifier and the SQUAD-EAT dataset on Github[20].

Below is an example from SQUAD-EAT with HUM:

Question: Which NFL team represented the AFC at Super Bowl 50?

Expected Answer Type: HUM.

5.3.5 Experiments and Results

Datasets

We experimented on the TrecQA dataset which is a standard dataset used to benchmark state of the art systems for answer sentence selection task. The authors of

[19] https://github.com/cmasch/cnn-text-classification
[20] https://github.com/rsanjaykamath/EAT-classifier

(Tayyar Madabushi et al., 2018; Madabushi and M. Lee, 2016) provide the EAT annotations for the TrecQA dataset based on their rule-based approach.

We modify the QA dataset SQUAD (Rajpurkar, J. Zhang, et al., 2016) designed for machine comprehension, into an answer sentence selection dataset to provide the answers in their original context. We name it as *SQUAD-Sent*. We do this by processing the dataset where each example is usually a triple of Question, Paragraph and Answer span (Text and the answer start offset in the paragraph) into a dataset where each triple is a Question, Sentence and Sentence label. The sentence label is 1 if the answer is present inside the sentence, else it is 0. We perform sentence tokenization using spacy toolkit[21] on paragraphs of SQUAD and perform a check for an exact match of answer strings in them. *SQUAD-Sent* is a special case dataset where there is just one positive sentence per question and the other sentences are negative examples. The motivation to do this is that we hypothesize that the *Reading Comprehension* task on SQUAD dataset might perform better when the input paragraph is a single sentence with an answer, instead of a paragraph with multiple sentences. Since the answer is present in only one sentence in a paragraph, reducing the paragraph size might increase the probability of finding the correct answer in the paragraph. In order to obtain the single sentence with an answer we can rely on *Answer Sentence Selection* task. Also since SQUAD dataset is a large scale human annotated dataset, we decided to modify this for the *Answer Sentence Selection* task. For the expected answer types of SQUAD questions, we use SQUAD-EAT which is a dataset with EAT annotated questions on SQUAD v1 dataset questions which is annotated by the authors of (Tayyar Madabushi et al., 2018; Madabushi and M. Lee, 2016) on our request.

WikiQA dataset by (Y. Yang et al., 2015) is another dataset used for answer sentence selection task which was built using Bing search engine query logs. We use a preprocessed version as used by (Rao et al., 2016) which has removed certain examples without any positive answers and questions with more than 40 tokens to compare the scores. The questions and answer sentences are annotated with EAT information as described in section 5.3.4.

Annotation of the entities in paragraphs

We detect the entities in the sentences using Dbpedia Spotlight tool by (Daiber et al., 2013). The detected entities by the spotlight are verified for their entity type match using the Spacy NER tool which is mapped to EAT using the mapping shown in table 5.6. Only the matching entities are highlighted and others are discarded.

[21] https://spacy.io

We also try using a named entity recognition tool to annotate entities in the text directly without using Dbpedia spotlight. Spacy[22] was used to annotate plain text and the entities matching the EAT were annotated.

Dataset	Split	#Plain Q	#EAT Q	#Entities
Trec QA	Train	1229	649 (52.8%)	13.96
	Dev	82	76 (92.68%)	5.02
	Test	100	82 (82%)	7.82
SQUAD-Sent	Train	87,599	78,740 (89.99%)	0.44
	Dev	10,570	9,606 (90.87%)	0.49
	Test	-	-	-
Wiki QA	Train	873	859 (98.39%)	0.15
	Dev	126	124 (98.41%)	0.03
	Test	243	236 (97.11%)	0.16

Tab. 5.10: Statistics of datasets with plain and EAT annotated questions. '#' refers to "Number of." **#Plain Q** is the number of questions in the whole dataset, **#EAT Q** is the number of questions which are annotated with EAT (a subset of #Plain Q), **#Entities** is the number of entities on average per question in the paragraphs for the **#EAT Q** question set - only those entities which match the EAT are annotated and not the rest.

Table 5.10 shows the statistics of the datasets with EAT annotated questions and plain word level questions (regular datasets) and the number of entities annotated in each set. EAT version of TrecQA dataset is as reported in (Tayyar Madabushi et al., 2018) and available through this link[23]. SQUAD-Sent was annotated by the authors of (Madabushi and M. Lee, 2016) upon our request. Wiki QA dataset although has high number of questions with a predicted EAT (the rest are *NO_EAT* class), the percentage of entities in the paragraphs which match the EAT are very low. This can be because of three reasons as explained earlier, 1) Undetected entities in the paragraph. 2) Wrongly predicted EAT. 3) Wrong taxonomy mapping.

Results

Table 5.11 shows various results on different versions of datasets. Note that the questions in the following experiments of Table 5.11 contain all the questions from the datasets, which includes questions which are highlighted with EAT and questions which are not highlighted with EAT as well. Note that we test our systems on the Raw version of TrecQA test dataset.

[22] https://spacy.io/

[23] www.harishmadabushi.com/research/answer-selection/

Datasets	Method	Acc.@1	MAP	MRR
TrecQA	Plain words - (Rao et al., 2016)	-	78	83.4
	EAT words - (Tayyar Madabushi et al., 2018)	-	83.6	86.2
	Plain words - RNN-S	78.95	80.24	84.81
	EAT words (single type) - RNN-S	85.26	85.28	**89.16**
	EAT words (different types) - RNN-S	85.26	**85.48**	88.11
	EAT words (MAX+different types) - RNN-S	**86.32**	85.42	88.86
SQUAD-Sent	Plain words - Implementation of model by (Rao et al., 2016)	-	-	58.08
	Plain words - RNN-S	83.94	-	90.5
	EAT words (single type) - RNN-S	84.21	-	90.65
	EAT words (different types) - RNN-S	**84.26**	-	**90.70**
	EAT words (MAX+different types) - RNN-S	84.24	-	90.69
WikiQA	Plain words - (Rao et al., 2016)	-	70.9	72.3
	Plain words - (Tymoshenko and Moschitti, 2018)	-	**75.59**	**77.00**
	Plain words - RNN-S	56.79	69.07	70.55
	EAT words (different types - NER) - RNN-S	55.14	66.56	68.10
	EAT words (MAX+different types - NER) - RNN-S	55.14	66.25	67.92
	EAT words (single type) - RNN-S	56.38	68.63	70.59
	EAT words (different types) - RNN-S	58.4	**70.04**	**71.56**
	EAT words (MAX+different types) - RNN-S	57.20	69.17	70.89

Tab. 5.11: Results reported on TrecQA, WikiQA, and SQUAD-Sent datasets. SQUAD-Sent dataset is a modified version for answer sentence selection task. RNN-S is RNN-Similarity model.

TrecQA

The current state of the art system is by (Tayyar Madabushi et al., 2018) that uses EAT on word level model of (Rao et al., 2016). Henceforth both results are presented. Our model RNN-Similarity on plain word level data fetches better result than the model of (Rao et al., 2016) by 2.24 % on MAP and 1.41 % on MRR. Our EAT words (single type), EAT words (different types) and EAT words (MAX + different types) models outperforms the state of the art performance for both MAP (1.68%) and MRR (2.96%) scores of the previous state of the art model by (Tayyar Madabushi et al., 2018) where the MAP and MRR scores are higher for correct sentences being ranked as Top-1. In this way, we obtained the best accuracy scores by integrating the occurrence of the EAT in the answer sentences.

WikiQA

WikiQA dataset was annotated by our EAT-Classifier model which predicts EAT for the questions. NO_EAT predicted type is excluded from the experiments. A recent model by (Tymoshenko and Moschitti, 2018) which uses kernel methods outperforms all the scores of our model. We note that the performance on our EAT level models is higher than the ones on plain words. Plain text input fetches 69.07% on our RNN-S model, whereas the entity highlighted input with different types of entities fetches

70.04% with a slight improvement. Only a few number of entities are annotated by spotlight compared to other datasets which is shown in the table 5.10. To annotate entities better we experimented using Spacy NER types (marked as NER) directly which resulted in more annotated entities but reduced the performance lower than the word level scores.

SQUAD-Sent

SQUAD official test set is hidden to the public users. Although the difference between word level and EAT word level is little, the difference highlights the fact that the entity words replaced in the sentence would not worsen the performance of the systems; instead it improves it subtly. We would like to note that the MAP and MRR values were the same because of the existence of just 1 positive sentence amongst other negative per question. Hence we only report MRR on this dataset. Plain words - (Rao et al., 2016) performance is obtained using the implementation available online[24], which we experimented on SQUAD-Sent dataset.

One aspect to be highlighted is that the implementation[24] of word level model by (Rao et al., 2016) originally made for TrecQA dataset performs poorly (58.05%) on SQUAD-Sent dataset (maybe because SQUAD-Sent has only one positive answer sentence per question whereas other datasets have several ones) which motivated us to build a model (RNN-Similarity) which works robustly for all the three datasets we have experimented with, without changing any specific hyperparameters of these models.

Datasets	Method	Acc.@1	MAP	MRR
TrecQA (EAT)	EAT words (single type)	84.15	84.81	87.17
	EAT words (different types)	85.37	85.45	88.18
	EAT words (MAX+different types)	**85.37**	**85.06**	**89.20**
SQUAD-Sent (EAT)	EAT words (single type)	83.81	-	90.53
	EAT words (different types)	84.04	-	90.61
	EAT words (MAX+different types)	**84.16**	-	**90.73**
WikiQA (EAT)	EAT words (single type)	**58.02**	**68.91**	**70.99**
	EAT words (different types)	55.14	67.70	69.52
	EAT words (MAX+different types)	56.38	68.16	69.83

Tab. 5.12: Results reported on TrecQA, SQUAD-Sent and WikiQA datasets using RNN-Similarity model trained only on EAT annotated questions

[24] https://github.com/castorini/Castor

Table 5.12 shows various results on TrecQA SQUAD-Sent and WikiQA datasets with only the questions which are annotated with EAT information in the train and test sets.

Training datasets contain questions which are annotated with EAT information, if the question does not have an EAT annotated, it is discarded from the dataset below are the set of experiments and results:

- TrecQA (EAT): EAT words (MAX + different types) version of the dataset fetches the best scores on the test set with only EAT annotated questions. The same experiment with same dataset performed by (Tayyar Madabushi et al., 2018) fetches lower results (MAP: 81.74% and MRR: 82.93%).

- SQUAD-Sent(EAT): EAT words (MAX + different types) version of the dataset fetches the best scores on this dataset.

- WikiQA (EAT): We remove the questions with 'NO-EAT' class which were 23 questions overall. The results are better with EAT (single type) which shows that the method works well in certain cases better than different types of EAT as in the above two datasets.

The results reported in table 5.12 show that there is not a high improvement over different methods when trained only on questions with EAT information. Henceforth it is better to train models with the entire dataset and highlight EAT information only when the question contains the EAT information.

Conclusion

We report our findings on TrecQA, SQUAD-Sent and WikiQA dataset performance and show that we outperform state of the art results on TrecQA dataset[25] by the two different ways of highlighting Expected Answer Types in the data compared to the plain text input.

The Expected Answer Types are a useful piece of information that used to be extensively exploited in the traditional QA systems. Using them with the current state of the art DNN systems for *Answer Sentence Selection* task improves the system performance. We propose a simple model using recurrent neural networks which works robustly on three different datasets without any hyperparameter tuning and annotate entities belonging to the expected answer type of the question. Our model

[25] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

outperforms the previous state of the art systems in the answer sentence selection task on TrecQA dataset. We also propose a model to predict the expected answer type based on the question words using a multiclass classifier trained on a rule based system's output on a large scale QA dataset.

In the above set of experiments, we observed that highlighting Expected Answer Types (EAT) in the sentences works better in the *Answer Sentence Selection* task. Therefore we proceed to experiment on the *Reading Comprehension* task, whose goal is to extract short answers from the answer sentences, by highlighting the entities.

5.4 Using Embeddings of Entities for the Reading Comprehension task

In the previous sections we discuss about highlighting entities with respect to *Expected Answer Types* for *Answer Sentence Selection* task. In the case of *Reading Comprehension task* the same technique cannot be applied in a straightforward manner because the answer tokens or words are substrings of the answer sentences and if an answer word is an entity then replacing it with generic entity words such as *entity_left* or *entity_hum* etc. will result in loss of information during evaluation. Henceforth a different approach with the same intuition is carried over in this set of experiments.

We propose to take into account entities using entity embeddings for *Reading Comprehension* task. Usually QA systems use pre-trained word embeddings trained using Glove (Pennington et al., 2014), Word2Vec (Tomas Mikolov, Sutskever, et al., 2013) or Fast-text models (Tomas Mikolov, Grave, et al., 2018) where entity words, stop words, other words are considered. Using special entity embeddings for entities have been shown to have better performance in certain state of the art methods for evaluating word embeddings (Sherkat and Milios, 2017), (Dhingra, H. Liu, et al., 2017), (Kamath et al., 2017b) and also in the above for *Answer Sentence Selection* task where entities are represented with their type as a special embedding.

Often in the case of *Reading Comprehension* task the entity themselves play a role in being the answer tokens therefore the same technique used in the above section cannot be used here. Each entity has to be represented by its own embedding. In order to learn specialized entity embeddings for the entities we chose to learn word and entities embeddings (by highlighting entities during embeddings pre-training) in the same embedding space. The intuition behind learning a representation of words and entities is that the textual data sources such as Wikipedia dumps, news data, common crawl data etc. often contain single or multi-word entities along with regular words which can be used to represent entities in a special manner along with other non-entity words. For example, phrases "New York City" and "New Orleans" contain the word "New" which has the same word embedding in both the phrases. Our work shall use "New York City" as one single word embedding as it is a multi word entity, same for "New Orleans" which will be a single word embedding. Our hypothesis is that, doing so would improve sentence and entity matching and a better association between the LAT word of the question and the entities without any supplementary annotations required.

We describe our solution for encoding the information related to entities in a Reading Comprehension QA model and show that using entity embeddings with word embeddings outperforms certain methods which use only word embeddings on factual questions.

Motivation

Open-domain question answering systems often consists of question-answer pairs where the answers are entities. As shown in Table 5.13 & 5.14, there is a high number of questions whose answers are entities and their paragraphs contain 10 entities on average. Question answering systems input tokens in general do not distinguish between regular words and entities. Apart from using input features such as named entities, pos tags etc. the input tokens still remain at word level or character level.

Data	#Ques	#Ques with Entities
Train Ques.	87599	61033 (69.67%)
Dev Ques.	10570	7359 (69.62%)

Tab. 5.13: Number of questions with at least one entity as answer, in SQUAD dataset

Data	#Documents	Avg # entities
Train Documents	87599	10
Dev Documents	10570	11

Tab. 5.14: Average number of entities in the SQUAD dataset paragraphs

We propose to use a joint representation of words and entities together in the text to better represent entities. Multi-word entities (an entity described using more than 1 word) are clubbed into one single entity token.

Joint Entity and Word embeddings

Concatenating a phrase of entity words into one token makes it easier to represent multi-word entities like *European Union Commission* and *European Union Law* into a richer contextual representation to distinguish between different entities. The QA system then will have to retrieve a single entity token rather than an N-gram of word tokens.

Encoding symbolic information such as knowledge bases with text documents has become a reliable way to work with such information. Word and entity embeddings are widely used nowadays in several NLP tasks. The former have shown that it is a great tool to capture both semantic and syntactic information about words (Tomas Mikolov, Sutskever, et al., 2013). Several works have been proposed for their combination (Fang et al., 2016; Yamada et al., 2016) and we opt for the recent EAT algorithm proposed by (Moreno et al., 2017) for its simplicity. This work jointly learns words and entities (*Entity* + *Word*) in a unique embedding space (*EW-Emb*) and restrict the entity usage to Wikipedia pages only. The underlying idea is to exploit entity mentions, or anchor texts, within Wikipedia pages to calculate the entity embeddings following (Tomas Mikolov, Sutskever, et al., 2013). When an anchor text is found in a window, EAT processes two times the same window, one for the word and another for the entity. In that way word embeddings are not downgraded and unambiguous entity embeddings are learnt. However, extra pre-processing steps are needed to ensure that anchor texts are not removed, entity entries in the vocabulary are identified, and entity entries are normalized to avoid redundancy (e.g., by applying all possible redirections in Wikipedia). These embeddings are used as input for a simple question answering system. Our implementation is based on an optimized version of Word2Vec using TensorFlow[26] with parameters (embedding size = 200, learning rate = 0.025, 5 epochs, Skipgram configuration, and window size of 5). The obtained vocabulary is composed by more that 5.2M entries including 1.8M entities.

The QA system

The QA system used in this set of experiments is *DRQA* by (Chen, Fisch, et al., 2017) whose model is explained in Section 4.1.1. The model remains exactly the same but we have made certain changes for the input words.

We transform the SQUAD dataset which is built using Wikipedia dumps as shown in the example below

[26] https://tinyurl.com/y8p4457e

Q: What is the main executive body of the EU?

P: The European Commission is the main executive body of the European Union.

A: European Commission

Ent.Q: What is the main executive body of the *Wikipage_European_Union?*

Ent.P: The *Wikipage_European_Commission* is the main executive body of the *Wikipage_European_Union.*

Ent.A: *Wikipage_European_Commission*

At input, we use word and entities embeddings for input tokens. The multi-word entities become a single token because of entity embeddings. This results in change of offsets and answer text for the answers in the paragraph which causes issues for *Exact Match* evaluations as done by *Reading Comprehension* systems. Therefore during the preprocessing time we build a mapping between multi-word entities to their individual word offsets in the plain text paragraphs to use this during the evaluation to perform evaluation on the official dev set answers provided by SQUAD. The evaluation however is the official evaluation of SQUAD task based on plain text entities.

An example of a QA pair is shown above, first box represents a regular QA pair, and second box represents a QA pair after pre-processing with entities. Also note that in this setting, the answer is always an entity. Offset refers to character level offset to the start of the answer span. Mapping refers to the entity answer in plain text which is represented by both start and end character offsets which is used during evaluation.

We chose SQUAD data for our experiments because it is built using Wikipedia articles, which are also used to train our embeddings. To annotate entities in SQUAD data, we use DBpedia spotlight (Daiber et al., 2013), an off-the-shelf tool for annotating mentions of DBpedia resources in text. We use redirections file from dbpedia [27] to use the latest Wikipedia page titles for the entities after the redirections.

It was not suitable to find the wikipage for SQUAD paragraphs and use entities from the wiki dump because Wikipedia does not annotate all entity occurrences in

[27] http://wiki.dbpedia.org/develop/datasets/downloads-2016-10

the paragraph. Using Dbpedia spotlight we can annotate all the occurrences of the entities in the text.

We evaluate on the official SQUAD dev set as the test set is not available for public testing. Along with the whole dataset we also test only questions whose answers contain entities (for *Entity question answering*).

Data	Official (All questions)	Pre-processed (Entity questions)
Train	86832	17407
Dev	10570	2705

Tab. 5.15: Number of questions in SQUAD dataset official vs pre-processed

To perform the evaluation of *Entity question answering*, we preprocess the SQUAD dataset - train and dev sets to retain only those questions whose answers only contain entities. Table 5.15 shows the dataset statistics before and after the pre-processing. Questions after pre-processing contained **18%** of *Who*, **11%** of *Which*, **62%** of *What*, **7%** of *Where* types of questions.

Experiments and Results

Data & Embeddings	EM	F1
Words | Glove	69.5	78.8
Words | EW-Emb	66.59	74.89
Ent+words | EW-Emb	65.76	75.05

Tab. 5.16: Whole SQUAD dataset system performance with dev set of 10570 questions. EW-EMB is the *entity+word embeddings*

Data & Embeddings	EM	F1
Words | EW-Emb	67.43	71.85
Words | Fast text	70.94	75.53
Words | Glove	72.90	77.35
Ent+Words | EW-Emb	**79.55**	**80.35**

Tab. 5.17: Entity QA system performance on pre-processed SQUAD dev set of 2705 questions. The same *EW-Emb* space does not perform well when used on only word (W) representations. Glove performs the best on only words (W) representations with comparable results on Fast text models.

We conduct experiments with three different embedding spaces:
1) EW-Emb (entity+word embeddings)
2) Global Vectors (Glove)

3) Fast text vectors

Paragraphs with words (W) and paragraphs with Entities + Words (E+W) on entity annotated SQUAD dataset questions and also on the whole SQUAD dataset questions. Table 5.16 represents the results on Whole SQUAD dataset questions and 5.17 represents the results on the entity annotated SQUAD dataset questions.

One of the goal is to evaluate the performance of the system on extracting entities from the text. We evaluate with exact match and F1 scores on the retrieved results. Table 5.16 shows that using entity annotations and embeddings causes the results to fall by around 1% on the whole SQUAD dataset in comparison with *EW-Emb* embeddings. This might happen because the answers often are phrases and are not tagged as entities. However the *EW-Emb* embeddings space does not perform well when used on only words, compared to Glove embeddings space. It is then difficult to get a true conclusion about using entity embeddings on this dataset.

In Table 5.17, the questions whose answers are entities perform much better with entity annotations and embeddings by 7% increase in performance over Glove embeddings. The Entity+Word representation performs the best with *EW-Emb* space because of presence of entity tokens with entity embeddings.

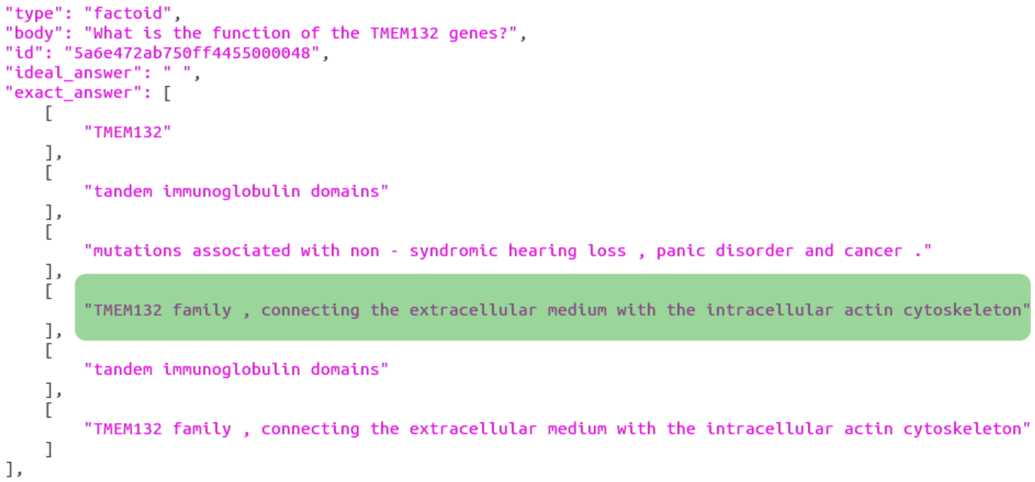
In SQUAD dataset, the answers are textual phrases sometimes containing just one entity (like in Entity QA setting) and sometimes containing an entity along with other surrounding contextual words which might not be entities but stop words, pronouns etc. (whole dataset setting). In some cases even short answers which are entities are not annotated by named entity recognition tools. Because of this phenomenon, Entity+Word embeddings perform better only when the answers are single entities which are short answers. Longer answer phrases are better answered with only word embeddings.

Conclusion

Entities are widely present in open domain textual corpus and using a jointly learnt embedding space benefits in better performance of QA with entities. We have reported our findings of using such embeddings with an open domain QA dataset which we have annotated with entities. Results show that the entity embeddings improve retrieval of entity answers over the regular word embedding spaces when the answers are entities. The performance does not improve but gets worse when the entities are not present in the answer spans.

In the above sections, we have discussed several ways of improving QA models which are mainly based on neural networks. These models focus only on getting better performance on their Top-1 accuracy scores. In the following section, we propose to post-process the output of neural network based QA models to further improve accuracy using semantic features from different paragraphs relative to a question.

5.5 Improving the QA Performance using Semantic and Structured Resources in a Ranking model



```
"type": "factoid",
"body": "What is the function of the TMEM132 genes?",
"id": "5a6e472ab750ff4455000048",
"ideal_answer": " ",
"exact_answer": [
    [
        "TMEM132"
    ],
    [
        "tandem immunoglobulin domains"
    ],
    [
        "mutations associated with non - syndromic hearing loss , panic disorder and cancer ."
    ],
    [
        "TMEM132 family , connecting the extracellular medium with the intracellular actin cytoskeleton"
    ],
    [
        "tandem immunoglobulin domains"
    ],
    [
        "TMEM132 family , connecting the extracellular medium with the intracellular actin cytoskeleton"
    ]
],
```

Fig. 5.7: An example of answer predicitons from BIOASQ data

Question Answering systems focus mainly on optimizing Top-1 accuracy of answers. Throughout our experiments with *DRQA* or *PSPR* model, we find that the Top-5 accuracy is always marginally higher than Top-1 accuracy. Figure 5.7 shows an example from the BIOASQ dataset predictions by the *OpenQA* model. The answer highlighted in Green is the correct answer for the question but it is present in Top-3 position.

To understand the importance of ranking Top-K predictions on the performance, we analyze Top-5 predictions on obtained from a QA model on a open domain dataset and a biomedical domain dataset. We first present the open domain analysis on SQUAD dataset (Rajpurkar, J. Zhang, et al., 2016) consisting of only wrongly answered questions based on the predictions by the DRQA model (Chen, Fisch, et al., 2017). We consider only the wrongly answered set to evaluate how many answers would have been correctly answered in the Top-5 predictions. Out of 3242 wrongly answered questions, 1433 questions had correct answers in Top-5 predictions. 44.2% of wrongly answered questions had a correct answer in Top-5. SQUAD dataset has

10,570 total dev questions, 13.55% (1433/10,570) of questions which are wrong in Top-1 predictions, contained an answer in Top-5 predictions.

We report a similar analysis on BIOASQ 5B (Tsatsaronis et al., 2015) dataset on the wrongly answered questions. We use DRQA model pre-trained on SQUAD dataset and fine-tuned to BIOASQ 5B dataset. Out of 98 wrongly answered questions in the overall test set, 35 questions had correct answers in Top-5 predictions. 35.71% of wrongly answered questions had a correct answer in Top-5. BIOASQ 5B dataset has 150 overall test questions (from 5 batches), 23.33% (35/150) of questions which are wrong in Top-1 predictions, contained an answer in Top-5 predictions.

The above statistics show that there is a scope of improvement to score better on Top-1 using only the Top-5 predictions from the model. We experiment with other values of Top-K by modifying K for ranking experiments. Only for the above analysis we use Top-5 predictions.

In the case of *Open Question Answering task*, due to the availability of several paragraphs which might contain the answer, the answer candidates might end up overlapping among different paragraphs answers. This overlap feature between these paragraphs has proven to be highly reliable for ranking answers in feature based QA models (Grappy, Grau, et al., 2011). The same kind of data (*Open Question Answering task* data with several paragraphs) is present in the BIOASQ task where the answers are contained in more than 1 paragraph and sometimes there are zero answers.

In order to take into account, the possibility of answer overlap among different paragraphs along with the other explicit semantic features we presented in the preceding sections, we define a set of features which are detailed in the section below and build ranking models based on Top-K answers predicted by the OpenQA system.

5.5.1 Modelling

In this study, we intend to rank the predictions of the chosen OpenQA model, therefore we present in detail its overall process.

The Paragraph Reader model extracts answer spans as shown in the *DRQA* model (Chen, Fisch, et al., 2017). The model gives two probabilities (one for start and one for end token given by two classifiers) as described in equation 5.7 and 5.8. The answer probability $Pr\left(a|q, p_i\right)$ for each answer span where p_i is i^{th} paragraph in Paragraph set P is computed as shown in the equation 5.9.

$$Pr_s(i) \propto \exp\left(\mathbf{p}_i \mathbf{W}_s \mathbf{q}\right) \tag{5.7}$$

$$Pr_e(i) \propto \exp\left(\mathbf{p}_i \mathbf{W}_e \mathbf{q}\right) \tag{5.8}$$

$$Pr\left(a|q,p_i\right) = \sum_j Pr_s\left(a_s^j\right) Pr_e\left(a_e^j\right) \tag{5.9}$$

The paragraph selector model gives a probability score $Pr\left(p_i|q,P\right)$ for each paragraph p_i in the paragraph set P, where **W** is a learnt weight matrix and the probability of each paragraph is calculated via a max-pooling and a softmax layer on question **q** and paragraph $\hat{\mathbf{p}}_i$ inputs as shown in the equation 5.10.

$$Pr\left(p_i|q,P\right) = softmax\left(\max_j\left(\hat{\mathbf{p}}_i^j \mathbf{W} \mathbf{q}\right)\right) \tag{5.10}$$

The softmax operation in equation 5.10 is applied over total number of paragraphs per question therefore a probability value is predicted for each paragraph.

Combining the two probabilities (from eq.5.9 and eq.5.10), the overall answer (Top 1) is chosen by choosing the highest probable answer from $Pr(a|q,P)$ for a question q which is calculated as :

$$Pr(a|q,P) = \sum_{p_i \in P} Pr\left(a|q,p_i\right) Pr\left(p_i|q,P\right) \tag{5.11}$$

To perform ranking over Top-K predictions, we need K predictions for a pair of question and a set of paragraphs. We use K as a hyperparameter and test with different number of K.

The model of *PSPR* returns a Top-1 answer based on the maximum combined probability (equation 5.11) computed by multiplying individual probabilities of paragraph (equation 5.10) and answer (equation 5.9) per paragraph and summed over all the paragraphs per question. In our case, we use Top-K answers using the same formula, instead of Top-1 we choose Top-K and choose paragraph which resulted in the prediction.

The goal is to improve the top 1 prediction by reranking the top K predictions of the model. We model the ranking task as a classification problem where only the top answer prediction of the training data is marked as *True* class and other top $K - 1$ predictions are marked as *False* class.

For the classification task we use 1) Random Forests, 2) Adaboost, 3) MLP classifier implementation using scikit learn[28].

5.5.2 Features for classifiers

We define features for the classifiers which are 1) Single features - features computed over a single paragraph. 2) Collective features - features computed over several paragraphs. 3) Semantic features for dealing with EAT or LAT verification.

The input features to the classifier models are described below.

Single Features

- **Answer Probability** - as computed in Equation 5.9 is the probability of answer spans obtained from the reader model (top answers have higher probabilities)

- **Paragraph Probability** - as computed in Equation 5.10 is the probability of paragraph obtained from the selector model (top scored paragraphs have higher probability to contain an answer)

- **Paragraph and Answer length in characters**

- **Answer words overlap with paragraph words** - Ratio of answer words overlapping with paragraph words. Length of $A \cap P$

Collective Features

- **Maximum value of answer probability** - Maximum answer probability for a prediction, over the number of paragraphs.

- **Maximum value of paragraph probability** - Maximum paragraph probability for a prediction.

[28] https://scikit-learn.org/

- **Answer presence ratio** - Count of number of predicted answer occurrences across different paragraphs, which is divided by the number of paragraphs.

- **Summation of answer probability** - If an answer prediction is repeated several times coming from different paragraphs, their probabilities are summed into one value.

- **Answer rank** - Rank of the predicted answer according to the model.

Semantic Features

- **EAT Match** - Expected Answer Type from the question matching the answer's entity type - '0' when there is no EAT match, '1' when there is an EAT match and '-1' when there is no EAT predicted in the question.

- **Cosine distance between LAT and answer - Semantic feature** - Lexical Answer Type words from questions and Answer prediction words are averaged and checked for cosine distance between them in Glove word embeddings.

For the feature **Cosine distance between LAT and answer**, the LAT words are computed using a classifier built using a CRF model named Wapiti[29] by (Lavergne et al., 2010) to determine which words in the question phrase are the LAT words. The task is a sequence labelling task where the labels are "1" or "0" to predict if a word belongs to a LAT or not. For the training dataset, we use the annotations provided by (Madabushi and M. Lee, 2016) which also highlights important words which contribute as LAT words.

For the feature **EAT Match** - we use the EAT Classifier explained in section 5.3.4, which predicts an Expected Answer Type for an input question. The model is trained on SQUAD EAT dataset provided by (Madabushi and M. Lee, 2016).

The same set of features (except the EAT Match) apply for both open domain and biomedical datasets. We define some additional ones for Biomedical domain as listed below.

- **Lexical Answer Type UMLS semantic Type match** - LAT word predicted using Wapiti is annotated with Metamap for UMLS semantic type which is checked for matching with Answer's UMLS semantic type.

[29] https://wapiti.limsi.fr/

- **Lexical Answer Type UMLS semantic Group match** - LAT word predicted using Wapiti is annotated with Metamap for UMLS semantic group which is checked for matching with Answer's UMLS semantic group.

- **Lexical Answer Type UMLS CUI match** - LAT word predicted using Wapiti is annotated with Metamap for CUI (Concept Unique Identifier) which is checked for matching with Answer's CUI.

The above features are marked as '0' when there is no match, '1' when there is a match and '-1' when there is no EAT predicted in the question.

5.5.3 Experiments and Results

Classifiers

We experiment with a few classifiers which takes as input, numerical features and predicts either *True* or *False* for input data. The scikit learn package provides several algorithms for binary classification, we experimented with 10 different classifiers mentioned in the post[30] which compares different algorithms. We report results on top 3 better performing classifiers - *Randomforests, Adaboost and MLP Classifier*.

Open Domain Data

Algorithm	Accuracy
PSPR - Baseline	41.1
Randomforest	42.86
Adaboost	**43.23**
MLP classifier	42.23

Tab. 5.18: Experiments on QUASAR-T dataset using different algorithms with best hyper parameters with all features Single, Collective and Semantic features from above. The best K value was found to be 3 on this dataset.

We experiment with QUASAR-T dataset which is an OpenQA dataset with several paragraphs per question. QUASAR-T dataset contains 37,012 training dataset questions and 3,000 dev and test dataset questions each. Results shown in Table 5.18 is conducted using all the features explained in the section above using three different algorithms. The results reported are chosen based on the best performing scores validated on the official dev set and tested on the official test set. For all the classifiers

[30] https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

mentioned in the table 5.18, the hyperparameter *Number of estimators* was tested with different values from 0 - 2000 and the best performing score on dev set was chosen and results on the test set are reported.

For each question, we consider the answers which are predicted as *True class* and consider Top-1 as the highest scored candidate. If there is no *True class* prediction for a question, we consider the *False class* prediction with lowest scored candidate.

Top-K	Accuracy
1 - Baseline	41.1
2	42.40
3	**43.23**
4	43.03
5	42.9
6	42.8
10	42.63

Tab. 5.19: Experiments on choosing *K* value for Top-K predictions using Adaboost algorithm with all features mentioned above on QUASAR-T dataset.

Adaboost algorithm performs better than the other two models and beats the baseline score of *PSPR* model by 2%. The baseline scores is computed on the predictions taken by the model implementation provided on the author's Github[31] page. The *K* value for Top-K predictions for the above set of experiments was **3**.

Results shown in Table 5.19 show that choosing *K* value as **3** fetches the best result on Adaboost algorithm. The more *K* value is increased, the lesser the accuracy because of imbalance in True and False classes for Top-1 data which leads to more negative samples.

Feature ablation

In order to understand which feature is more important compared to the rest, we do a feature ablation test by using scikit learn feature importance plot and the list below represents the 5 most important features in decreasing order of magnitude.

1. **Maximum value of paragraph probability** - Max Para. Prob. - 28.46%

2. **Answer presence ratio** - 27.69%

3. **Summation of answer probability** - 15.38%

[31] https://github.com/thunlp/OpenQA

4. **Paragraph Probability** - Sent. Prob. - 09.23%

5. **Answer Probability** - 06.15%

Features removed from the best model	Accuracy
Best model score	43.23
- Summation of answer probability	43.03
- Answer presence ratio	42.53
- Maximum value of paragraph probability	42.23
- Sentence Probability	43.2
- Answer Probability	43.06
- (Summation of Ans Prob. + Answer presence ratio)	42.46
- (Maximum value of paragraph probability + Answer presence ratio)	41.36

Tab. 5.20: Experiments on Feature ablation and importance of features. Computing decrease in performance when certain features are removed from the input

To understand the importance of input features, we remove some features listed in the 5 most important features from above, and run the experiment to compute the decrease in performance. Results shown in Table 5.20 highlight the importance of choosing right features to increase performance. These features are important and sufficient to model the complexity of the input. Removing some features penalizes the performance. The most impactful features according to the above experiments were *Maximum value of paragraph probability* and *Answer presence ratio*, removing these two features drops the performance by 1.87%.

BIOASQ dataset

BIOASQ task expects Top-5 answers per questions to evaluate *Strict Accuracy (Top-1)* and *Lenient Accuracy (Top 5)*. Therefore in this case, we use Top-K as Top-5 for this purpose. That is why Lenient Accuracy remains the same in all experiments. In the previous set of experiments on open domain data in table 5.19, we found that choosing Top 3 answers performed with better results for Top 1 ranking, and the performance decreased upon increasing K value because of increasing number of negative answers in the data. Therefore we do not increase above 5.

In this set of experiments, the baseline system is the output from *PSPR* model evaluated using official BIOASQ evaluation. From the results it is clear that the Randomforest algorithm performs better when comparing with the rest. We evaluate this the same way do with QUASAR-T dataset by considering a dev set to check for the best hyper parameter and later apply it on the official test sets.

Datasets	Finetune	BIOASQ 4	BIOASQ 5	BIOASQ 6
Baseline	Strict	30.31	46.83	42.79
	Lenient	45.00	52.66	53.41
Adaboost	Strict	**39.37**	44.00	**45.96**
	Lenient	45.00	52.66	53.41
Randomforest	Strict	**38.75**	46.00	**46.58**
	Lenient	45.00	52.66	53.41
MLP	Strict	**34.37**	46.00	38.50
	Lenient	45.00	52.66	53.41

Tab. 5.21: Experiments using different algorithms with best hyper parameters with all features listed above for biomedical domain

Results shown in table 5.21 highlight the gain in the performance of Strict Accuracy scores. For BIOASQ 4 and BIOASQ 6 datasets, there is an increase in performance, but for BIOASQ 5 there is a small decrease in performance.

Feature ablation is important to understand which feature plays important role in the classification. Here are the few important features as calculated using scikit learn feature important plot. The list below represents the 5 most important features in decreasing order of magnitude.

1. **Maximum value of answer probability** - 21.87%
2. **Answer words overlap with paragraph words** - 15.28%
3. **Paragraph Probability** - 11.72%
4. **Lexical Answer Type UMLS semantic Type match** - 10.06%
5. **Lexical Answer Type UMLS CUI match** - 09.82%

Features removed	Finetune	BIOASQ 4	BIOASQ 5	BIOASQ 6
Best model score	Strict	38.75	46.00	46.58
	Lenient	45.00	52.66	53.41
- Max. Ans Probability	Strict	38.75	43.33	44.72
	Lenient	45.00	52.66	53.41
- Answer overlap	Strict	38.75	45.33	45.96
	Lenient	45.00	52.66	53.41
- LAT Semantic Type	Strict	38.75	45.33	45.96
	Lenient	45.00	52.66	53.41
- Paragraph Probability	Strict	38.75	44.66	45.96
	Lenient	45.00	52.66	53.41
- (Feature 1 and 2)	Strict	37.5	42.66	44.72
	Lenient	45.00	52.66	53.41

Tab. 5.22: Experiments on Feature ablation and importance of features.

The most impactful features according to the above experiments were *Maximum Answer Probability* and *Answer words overlap with paragraph words*, removing these two features drops the performance by more than 1% on average on all the three datasets. The semantic feature of LAT UMLS semantic type match has an impact on the classifier performance which shows a decrease in performance when removed.

Conclusion

With the above set of experiments, we try to rank the predictions obtained from a neural network to further improve the accuracy. In the case of open domain data and two sets of biomedical domain datasets, the traditional ML classifiers work better and improve the scores up by some margins, which gives an opportunity for improving results of deep learning models.

5.6 Applying the Open QA model on different datasets

Question answering systems have evolved over time, the former pipeline systems focused on answering a question by performing document retrieval, followed by paragraph ranking, and answer extraction. And currently individual task based systems which focus on individual tasks such as paragraph ranking or answer extraction have emerged. The goal always has been to build end-to-end systems where one model does it all, but since it is a hard task to tackle for a single model, smaller goal oriented systems are built.

Since there are different models for different individual tasks, different datasets with different task goals exist. In this section, we try to use existing datasets which were primarily built for a different task in a way that the pipeline approach would fit and show more realistic version of the task implementation on the datasets.

For example, SQUAD dataset by (Rajpurkar, J. Zhang, et al., 2016) assumes that the relevant paragraphs always exist for a question and the systems built on this dataset should comply to this assumption. But generally getting relevant paragraphs itself is a hard task which has been researched for decades. Therefore only *Open Question Answering* scenario is feasible as some paragraphs may not be relevant. Therefore we try to use SQUAD in such a way that it would fit as a *Open Question Answering* task.

5.6.1 OpenQA on SQUAD dataset

One way to use a *Reading Comprehension* task dataset as a *OpenQA* dataset is by completely ignoring paragraphs and using an IR engine to fetch the paragraphs and using it as input as done by (Chen, Fisch, et al., 2017). In this case, the IR engine retrieval was not based on a machine learning model but used as an off-the-shelf tool. Another way to build an end-to-end open domain question answering model is to modify the retrieval phase as a weakly supervised task and build an end-to-end with a single large scale model as done by (K. Lee et al., 2019) using the BERT model.

In our work, we consider the SQUAD dataset and the paragraphs, and split the paragraphs into sentences. In this setting we will only have one sentence with a correct answer and all other sentences will be marked negative as they do not have an answer. We hypothesize that determining a relevant sentence and extracting answers from it will be better than extracting answers from a lengthy paragraph which contains irrelevant sentences to the question. We call this dataset as SQUAD-SENT dataset.

SQUAD-SENT dataset has been experimented on *Answer Sentence Selection* task in section 5.3. In this section we apply the SQUAD-SENT dataset on *OpenQA* task for answer extraction.

Model	Accuracy
SQUAD-RC by (Chen, Fisch, et al., 2017)	69.5
SQUAD-SENT on PSPR	50.10
SQUAD-OPEN by (K. Lee et al., 2019)	26.5
SQUAD-OPEN by (Chen, Fisch, et al., 2017)	27.1
SQUAD-OPEN BM25+BERT by (K. Lee et al., 2019)	28.1

Tab. 5.23: Results on SQUAD variant dev datasets in Open Domain Question Answering and Reading Comprehension settings

The results presented in Table 5.23 are on different variants of SQUAD datasets but evaluated for the same answer extraction task as done in official SQUAD evluation setting. *SQUAD-RC* by (Chen, Fisch, et al., 2017) is the *DRQA* model scores on the SQUAD dataset with correct paragraphs.

SQUAD-SENT is a sentence level dataset which contains 1 correct sentence with an answer and rest are wrong sentences. The results presented above on *SQUAD-SENT* dataset is on the model of *PSPR* by (Y. Lin et al., 2018).

SQUAD-OPEN is an open setting where no paragraphs are given and the model itself uses some techniques to first retrieve a paragraph and then perform answer extraction. The results presented above by (K. Lee et al., 2019) uses a BERT model approach and learns the paragraph retrieval and answer extraction end-to-end using a weakly supervised approach. The *DRQA* model by (Chen, Fisch, et al., 2017) on the *SQUAD-OPEN* dataset setting use an answer retriever and answer reader in cascading fashion. Their model fetches better result than BERT (K. Lee et al., 2019) on *SQUAD-OPEN* data setting. The authors of (Chen, Fisch, et al., 2017) implement their own IR model (based on TF-IDF weighted bag-of-word vectors and include bi-gram features) without using machine learning for fetching relevant paragraphs.

An important highlight from these experiments is that the performance of the state-of-the-art model BERT (Devlin et al., 2018) on SQUAD v1.0 dataset is 87%, and the same model modified to SQUAD-OPEN dataset setting which does not provide a relevant paragraph to the model fetches 26.5% (K. Lee et al., 2019) on an end-to-end OpenQA BERT model. It is the similar case even for the DRQA model (Chen, Fisch, et al., 2017) which fetches 69.5% on SQUAD v1.0 dataset whereas the same model fetches 27.1% on SQUAD-OPEN dataset setting. This shows the wide gap in performance drop when the relevant paragraphs are not provided to the models.

Conclusion

The three different approaches with the above results show how different models trained on SQUAD dataset (some in a realistic setting without always containing a positive paragraph) cripples upon not having always some positive samples. As discussed in Chapter 4 and highlighted by (Talmor and Berant, 2019), current state-of-the-art deep learning models for QA are often overfitting on the task datasets and failed to generalize. In this set of experiments we show that these task datasets which are designed with specific goals do not perform similarly when some assumptions do not apply.

End-to-end models for overall Question Answering (OpenQA) seems to under perform even when pre-trained models like BERT are used. This shows the complexity of the overall QA task (fetching relevant paragraphs and extracting answers together). Task specific models like *Reading Comprehension* tasks are simplified QA tasks whose performance cannot be directly compared to OpenQA tasks.

5.6.2 TrecQA as OpenQA task

TrecQA task had been a widely popular QA task in the past that has released several small scale datasets released during 1999-2004 that are in different formats. From 2007 onwards, (M. Wang, Smith, et al., 2007) created and used these datasets for *Answer Sentence Selection* task and a lot of works[32] followed after that used this dataset for the same task.

TrecQA task organizers originally released a regex pattern answer set for the QA datasets which were used to judge the correct answers automatically. We use this pattern set to obtain the answer spans for the paragraphs in the well known TrecQA-13 (2004) dataset by (M. Wang, Smith, et al., 2007). We use the same regex patterns to determine the answer span from the answer paragraphs. We create a dataset with answer paragraphs and answer spans from the paragraph if they are relevant, if the paragraphs are not relevant then there will not be an answer in it. We call this dataset as *TrecQA-RC* and we release this publicly[33]. The number of questions and paragraphs are the same as the one created by (M. Wang, Smith, et al., 2007) but contains a binary label signifying the relevance of the paragraphs along with added information.

We apply a QA model on this dataset to have a baseline score. Since TrecQA dataset has both relevant and irrelevant snippets, it is more suitable for *OpenQA* task to be applied, so we apply the model of *PSPR* on this dataset to report the performance.

Experiments and Results

TrecQA train dataset ($\sim$1,000 questions) is relatively very small compared to the scale of QUASAR-T dataset ($\sim$40,000 questions) which is another dataset of the same type. We experiment by training a model just on the TrecQA dataset and predict on the test set. We also experiment using the pre-training and fine-tuning approach by first training on QUASAR-T dataset and later finetuning on the TrecQA dataset.

In table 5.24, we compare the previous state-of-the-art model results which used non neural network approaches (Severyn and Moschitti, 2013; Yao et al., 2013a) which report results on answer extraction task on TrecQA dataset without releasing the dataset for this task. Both the models fetch better scores by atleast 9% higher than the neural network models which we detail below.

[32] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)
[33] https://github.com/rsanjaykamath/trecqa-rc

Model	Accuracy
Sequence tagging model by (Yao et al., 2013a)	67.2
Tree kernels methods by (Severyn and Moschitti, 2013)	70.8
Trec only	45.68
Pretrained with QUASAR-T	49.38
RC mode	58.02

Tab. 5.24: Results on using TrecQA dataset for OpenQA task

We use the OpenQA model *PSPR* with TrecQA dataset for all experiments with slight variations. *Trec only* experiment is trained and tested only on TrecQA dataset using the *PSPR* model. Since the TrecQA dataset is small scale, we use a "pretrain and finetune" method. *Pretrained with QUASAR-T* experiment is pre-trained firstly on QUASAR-T dataset and fine-tuned with TrecQA dataset.

Training a DNN model for the small scale TrecQA dataset although fetches relatively similar results to compared to the pre-trained ones with QUASAR-T, it is always better to pre-train when there is a large scale dataset with similar task at disposal.

Our model of *RNN-SIMILARITY* (Kamath et al., 2019) on *Answer Sentence Selection* task has results of 85.2% MAP on TrecQA dataset for selecting the right answer sentence. But using a similar RNN model like *PSPR* on OpenQA task which uses paragraph selection and extraction module fetches around 45.68% results which is low for the answer extraction objective.

The assumption of always having an answer in the paragraph in datasets like SQUAD v1.0 gave rise to a huge amount of DNN based models on *Reading Comprehension* task. We experiment with the same assumption by having only relevant paragraphs (paragraphs with answers) to perform answer extraction. We term it as the *RC mode* where the paragraphs always contain an answer. We use the model of DRQA on this dataset. As expected from the previous experiments above, the *Reading Comprehension* - *RC mode* fetches the better scores than OpenQA mode on the TrecQA answer extraction as all the snippets are relevant.

The above results highlight that using neural network models does not always necessarily fetch better results than the non neural network models.

5.7 Conclusion

Specific domains, like biomedical domain, have plenty of resources handcrafted or generated or annotated by human experts. Using this along with plain text corpus for question answering is a challenging task and is not trivial. In this chapter, we addressed one of our research questions on leveraging these expert curated knowledge sources and semantic information effectively into state-of-the-art question answering models on biomedical domain and open domain systems (with other types of information apart from free text) to improve their performances.

Our goal was to utilize the availability of different sources of information and tools to enrich the plain text datasets with additional information for question answering. We believe that extracting features from text using knowledge sources and inputting them in different ways to deep learning models will have some positive impact. Since deep learning models are not best suited for small scale datasets, results on methods involving pre-training and fine-tuning can further improve scores if domain specific information is used in correct ways.

We highlighted the problem of lacking answer variants in biomedical domain dataset BIOASQ and manually annotated answer variants to show the significant difference the same model fetches upon annotating the answer labels correctly. We also proposed a method to automatically generate these variants using Metamap tool which fetched similar or even better results in some sets compared to manually annotated answers.

We presented the use of *Expected Answer Types* in question answering by verifying the presence of it in the answers. In both biomedical domain data and open domain data, there is a scope of improvement of QA model performance because there is a significant amount of wrongly answered questions whose answer types match *Expected Answer Types* from questions.

We used the *Expected Answer Types* to highlight entities in plain text by using a special method of annotation and entity embeddings to input this information. These methods for open domain works well for *Answer Sentence Selection* which scores better than merely on plain text data of TrecQA dataset. This method however works better on *Reading Comprehension* task only when the answers consists of entities and not long text phrases, the *Reading Comprehension* model performs better at finding the right answer entities on SQUAD dataset.

Since deep learning models almost always focus towards building end-to-end systems, not much emphasis is put towards post processing of the outputs to better rank the

Top-K predictions. We used some traditional machine learning binary classification models to rank a better answer candidate into Top-1 position and showed that there was a scope for improvement on the predictions from the neural network models. This applies both on biomedical question answering and open domain question answering models on *Open QA* task.

We found that some datasets like SQUAD which focuses on *Reading Comprehension* task performs poorly when the relevant paragraphs are removed and the task is modified into an *OpenQA* task. We applied *OpenQA* models on SQUAD dataset which is modified as a sentence level *OpenQA* task and showed the performance drop. We created an *Open QA* version of TrecQA task by adding answer spans from the paragraphs which can be extracted using the answer patterns provided by Trec. We presented some baselines scores using *PSPR model* and *Reading Comprehension* models on this dataset.

Inspite of using state-of-the-art massive language models like BERT, the performance on *OpenQA* tasks such as SQUAD-OPEN as shown by the authors of (K. Lee et al., 2019) is very low (26.5 on the end-to-end BERT model). This highlights the low performance on overall QA task with current state-of-the-art models.

Our publications related to the work described in this chapter are listed below:

- **2019 - Predicting and Integrating Expected Answer Types into a Simple Recurrent Neural Network Model for Answer Sentence Selection.** - Sanjay Kamath, Brigitte Grau, Yue Ma. 20th International Conference on Computational Linguistics and Intelligent Text Processing - CICLING 2019, April 2019.

- **2018 - An Adaption of BIOASQ Question Answering dataset for Machine Reading systems by Manual Annotations of Answer Spans.** - Sanjay Kamath, Brigitte Grau, Yue Ma. Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering. EMNLP, October 2018.

- **2018 - Verification of the Expected Answer Type for Biomedical Question Answering.** - Sanjay Kamath, Brigitte Grau, Yue Ma. HQA workshop, companion proceedings of the The Web Conference 2018, April 2018.

- **2017 - A Study of Word Embeddings for Biomedical Question Answering.** - Sanjay Kamath, Brigitte Grau, Yue Ma. 4e édition du Symposium sur l'Ingénierie de l'Information Médicale, November 2017.

6 Conclusion and Future Perspectives

In the context of GoASQ project[1], our goal was to investigate, compare and combine different approaches for answering questions formulated in natural language over textual data using semi-structured and structured data.

We introduced question answering by defining and explaining the general process, different types of tasks, different kinds of data used. Different QA systems are built based on different types of tasks. In this thesis, we detailed all the above aspects for the three tasks: *Reading Comprehension*, *Answer Sentence Selection* and *Open Question Answering*, and report experiments on several datasets on these tasks.

One of our research focuses was towards building better QA systems for domain specific datasets like biomedical domain data, although open domain is also studied on certain tasks, such as answer sentence selection. We listed some of the hurdles we encountered while building a deep learning based QA system using deep learning models and our research questions were aligned towards addressing them:

1. How can we build models which work both on small scale and large scale datasets without dropping performance?

2. How can we leverage semantic and structured knowledge effectively into state-of-the-art question answering models?

We found that a *hybrid QA system* that can deals with these two aspects worked the best for the QA task in biomedical domain - the BIOASQ task. In the chapter 4 we proposed to model the task as an *OpenQA* task which performed the best and outperformed the state-of-the-art models which model BIOASQ as a *Reading Comprehension* task. We showed that pre-training on multiple *Reading Comprehension* datasets obtained the best performance scores on a state-of-the-art model for BIOASQ. In the chapter 5, we proposed a method to annotate the answer variants in the training and test sets automatically which improved results by a large margin. This constitutes a realistic evaluation of the system performance. We also proposed to

[1] https://goasq.lri.fr/

use using semantic and structured information features from different paragraphs and showed that it improved the performance on the BIOASQ task.

In summary, to build a best performing model for BIOASQ QA task, we conclude that one can use multiple datasets in combination to pre-train the model, and fine-tune it on all answer variants which can be annotated automatically. The predictions from the above process can be further improved for better accuracy using the ranking model which uses semantic and structured information features at the post-processing phase.

In the following, we summarize the hybrid QA system details from both hybrid model and hybrid data point of view and give some future perspectives.

Building Models for Small Scale and Large Scale Datasets

The chapter 4 presents our work on building better models for question answering which suits both small scale and large scale datasets. The definition of *Hybrid Data* in this context refers to building models which are trained on different datasets from different domains to perform well on QA tasks.

Using deep learning models on small scale datasets will not fetch optimal results. One of our first goals was to adapt deep learning models to work on small scale datasets effectively. Therefore we began by choosing a model which worked well on open domain QA dataset and adapted it to biomedical domain. This process is called as domain adaptation. We formally defined domain adaptation which we use extensively in many experiments. We adapted a RNN based QA model which was built on open domain datasets such as SQUAD, towards biomedical dataset BIOASQ. We show the importance of doing domain adaptation by comparing results on models with and without domain adaptation.

According to this study, we believe that the domain adaptation will play a major role for small scale and domain specific datasets. Mainly because creating large scale datasets is not an easy task and it is more difficult and expensive when the data is domain specific. Domain adaptation and transfer learning are the two most important research disciplines to facilitate current open domain models to be used on small scale datasets and different tasks.

While performing domain adaptation, we came across another way of modelling the BIOASQ task, the OpenQA task, open domain question answering. It turns out that OpenQA is more suitable for BIOASQ task than Reading Comprehension task because of the nature of BIOASQ data which contains relevant and irrelevant paragraphs in the data relative to the gold standard answers provided. We compared two RNN based models on these two tasks and adapt these models to biomedical domain. We found that OpenQA model of PSPR better performed than the model of Reading Comprehension task, which was the DRQA model. We presented some factors which we considered as important before choosing a model to experiment and explained why we did some of our choices of models.

Several ways of modelling reading comprehension task have been proposed lately from works such as HotpotQA (Zhilin Yang, Qi, et al., 2018) and DROP (Dua et al., 2019) which point out the simplicity of the original task defined by SQUAD dataset (Rajpurkar, J. Zhang, et al., 2016). Harder QA tasks and datasets create a demand for new models which focus on tackling these problems.

While performing above experiments, we wondered about pre-training the models on different datasets. Therefore we experimented with different Reading Comprehension datasets by keeping the same model architecture and hyperparameters. We showed the variability in the performance on downstream domain adaptation when the initial models are trained with different datasets. SQUAD 2.0 dataset by (Rajpurkar, Jia, et al., 2018) was the best performing single dataset for pre-training the models. A combination of 4 Reading Comprehension datasets for pre-training performed the best on the fine-tuning for BIOASQ dataset. For the above experiments, we used BERT model by (Devlin et al., 2018) which was the state-of-the-art model while experimenting. We also compared different pre-trained BERT models and found BIOBERT (J. Lee et al., 2019) to be the best model for biomedical domain adaptation.

More data leads to better performance has been proven for biomedical domain adaptation in our experiments. In the future, we believe that more datasets should be built by explicit mention of possible biases attached with the datasets to make more people aware about the consequences.

Towards the goal of building end-to-end models which do not use explicit features but perform feature extraction themselves, less emphasis is put on classical techniques which have addressed the fundamental problems of question answering and have proposed certain useful features and methods to handle certain types of data. More emphasis should be put on creating diverse datasets, careful analysis of bias, and most importantly highlight where the model fails to fetch correct answers which provides insights on improvements required for future work. But the research articles published in the recent times do not address any negative results from the models, and also seldom do error analysis.

Leveraging Structured and Semantic Information into Question Answering Models

The Chapter 5 presents our work on leveraging semantic and structured knowledge effectively into state-of-the-art question answering models on biomedical domain and open domain systems to improve their performances.

Our goal was to utilize the existing information from different sources and tools to enrich plain text datasets. Our hypothesis was that the deep learning based QA model can be further improved using semantic and structured information from external sources pertaining to the input text. The definition of *Hydrid Data* in this context refers to using open domain data with domain specific data in a domain adaptation process, plus integrating structured knowledge for annotating training datasets and for enriching the input data. The definition of *Hybrid model* in this context refers to an addition of a post processing reranker to account for structured knowledge and collective features obtained from different paragraphs.

In this regard, we highlighted the problem of lacking answer variants in biomedical domain dataset BIOASQ and showed a large performance difference when all the answer variants are annotated and when the model learns from such annotated data. We provided manual annotations for these answer variants and proposed an automatic method which uses UMLS meta-thesaurus to generate these annotations which fetched similar results. We release the manual annotations online[2] along with the code to generate automatic variants[3].

In terms of future perspectives on the BIOASQ task and the dataset, we believe that it is important to focus research on providing better annotations, proposing guidelines for people from other domains, and performing fine grained analysis on the dataset instead of just focusing on building better models which improves accuracy.

We presented the use of Expected Answer Types in question answering by verifying the presence of it in the answers. In both biomedical domain data and open domain data, there is a scope of improvement of QA model performance because there is a significant amount of wrongly answered questions whose correct answer type match as of the Expected Answer Type. On open domain, we presented some statistics of SQUAD dataset questions containing Expected Answer Types as predicted by a model of (Madabushi and M. Lee, 2016) and on the biomedical domain, we presented some statistics on the BIOASQ dataset questions containing Lexical and Expected

[2] https://zenodo.org/record/1346193#.W3
[3] https://github.com/rsanjaykamath

Answer Types annotated by (Neves and Kraus, 2016) to support the above study on verification.

We use the Expected Answer Types to highlight entities in plain text by using a special method of annotation and special embeddings to input this information. These methods for open domain work well for Answer Sentence Selection. We release the code for the model used in this experiment[4]. This method however works better on Reading Comprehension task only when the answers consists of entities and not long text phrases. The Reading Comprehension model performs better at finding the right answer entities on SQUAD dataset. In the context of Expected Answer Types from input questions, we released the model[5] which predicts an Expected Answer Type for input questions.

In terms of future perspectives we believe that neural models must better integrate entity representations of knowledge coming from free text as well as from structured resources like ontologies, knowledge bases. Works such as (Ferré, 2019) align two different types of vector representations that capture part of their meanings with entities and text in the form of word embeddings and ontology concepts in the form of concept embeddings. The alignment is done in a supervised manner. This can be further extended towards complex tasks like question answering where the model integrates the knowledge from two sources while learning.

Since deep learning models almost always focus towards building end-to-end systems, not much emphasis is put towards post processing of the outputs to further improve predictions. We use some traditional machine learning models like randomforests, adaboost, multilayer perceptron etc. for binary classification to rank a better answer candidate into Top-1 position and show that there is a scope for improvement on the predictions from the neural network models. This applies both on biomedical question answering and open domain question answering models on Open QA task.

Using classical ML techniques to improve predictions from deep learning models are advantageous in the following ways: 1) Cheaper compute costs to improve predictions than train a complex model. 2) Faster inference times. These classical ML models might not be applicable for the overall QA task, but does improve scores when used on the outputs of a deep learning model to further improve scores.

In the analysis of (Talmor and Berant, 2019), the authors point out the overfitting of models to particular task and datasets they are trained. We find that some datasets like SQUAD which is focused on Reading Comprehension task performs poorly when

[4] https://github.com/rsanjaykamath/RNN-Similarity

[5] https://github.com/rsanjaykamath/EAT-classifier

the relevant paragraphs are removed and the task is modified into an OpenQA task. We apply OpenQA models on SQUAD dataset which is modified as a sentence level OpenQA task and show the performance drop. We create an Open QA version of TrecQA task by adding answer spans from the paragraphs which can be extracted using the answer patterns provided by Trec. We present some baselines scores using PSPR model and Reading Comprehension models on this dataset. We release this dataset online[6]. An end-to-end model for the OpenQA task was first proposed by Lee et al. (K. Lee et al., 2019), which learns the retrieval and the extraction phase together. This is one of the first works which aims at building an end-to-end model for the overall QA task. The low results obtained using the latest state-of-the-art model shows the real complexity of the overall QA task which cannot be easily addressed using a single end-to-end deep learning model including the latest pre-trained language models like BERT.

While we performed the majority of the above set of experiments, the contextual language models such as BERT by (Devlin et al., 2018) had not been released. Ever since then, the large scale pre-trained language models (LM) have changed the way most of the NLP tasks are addressed today. Earlier models used CNNs or RNNs with attention mechanisms for many tasks which are now being replaced by these large scale LM models like BERT. We believe these new models have a large scope for various NLP tasks and the traditional ML models like the classifiers we use (randomforests, adaboost etc.) still hold good in certain cases where the dataset size is small scale and might require domain expertise.

An end-to-end OpenQA model (K. Lee et al., 2019) which uses these pre-trained contextual language models failed to obtain equivalent scores compared to LSTM based models that use a pipeline approach. This shows the complexity involved in building end-to-end models for overall OpenQA task. We believe that an hybrid modelling approach is the way forward to build better systems on overall question answering task.

[6] https://github.com/rsanjaykamath/trecqa-rc

Bibliography

Abacha, Asma Ben and Pierre Zweigenbaum (2015). “MEANS: A medical question-answering system combining NLP techniques and semantic Web technologies”. In: *Information processing & management* 51.5, pp. 570–594 (cit. on p. 119).

Allam, Ali Mohamed Nabil and Mohamed Hassan Haggag (2012). “The question answering systems: A survey”. In: (cit. on p. 38).

Azad, Hiteshwar Kumar and Akshay Deepak (2019). “Query expansion techniques for information retrieval: a survey”. In: *Information Processing & Management* 56.5, pp. 1698–1735 (cit. on p. 40).

Beltagy, Iz, Arman Cohan, and Kyle Lo (2019). “Scibert: Pretrained contextualized embeddings for scientific text”. In: *arXiv preprint arXiv:1903.10676* (cit. on p. 98).

Ben Abacha, Asma, Chaitanya Shivade, and Dina Demner-Fushman (Aug. 2019). “Overview of the MEDIQA 2019 Shared Task on Textual Inference, Question Entailment and Question Answering”. In: *Proceedings of the 18th BioNLP Workshop and Shared Task*. Florence, Italy: Association for Computational Linguistics, pp. 370–379 (cit. on p. 63).

Berant, Jonathan, Andrew Chou, Roy Frostig, and Percy Liang (2013). “Semantic parsing on freebase from question-answer pairs”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544 (cit. on p. 29).

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1994). “Signature verification using a" siamese" time delay neural network”. In: *Advances in neural information processing systems*, pp. 737–744 (cit. on p. 44).

Chen, Danqi, Jason Bolton, and Christopher D Manning (2016). “A thorough examination of the cnn/daily mail reading comprehension task”. In: *arXiv preprint arXiv:1606.02858* (cit. on p. 47).

Chen, Danqi, Adam Fisch, Jason Weston, and Antoine Bordes (2017). “Reading Wikipedia to Answer Open-Domain Questions”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879 (cit. on pp. 11, 49, 54, 55, 69, 77, 78, 81, 83, 91, 93, 112, 120, 131, 141, 145, 146, 155, 156).

Choi, Eunsol, Daniel Hewlett, Jakob Uszkoreit, et al. (2017). “Coarse-to-fine question answering for long documents”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 209–220 (cit. on p. 55).

Proceedings of the Workshop on Machine Reading for Question Answering (July 2018). Melbourne, Australia: Association for Computational Linguistics (cit. on p. 64).

Chu-Carroll, Jennifer, James Fan, BK Boguraev, et al. (2012). “Finding needles in the haystack: Search and candidate generation”. In: *IBM Journal of Research and Development* 56.3.4, pp. 6–1 (cit. on pp. 2, 119).

Clark, Christopher and Matt Gardner (2018). “Simple and Effective Multi-Paragraph Reading Comprehension”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 845–855 (cit. on p. 58).

Collobert, Ronan, Jason Weston, Léon Bottou, et al. (2011). “Natural language processing (almost) from scratch”. In: *Journal of machine learning research* 12.Aug, pp. 2493–2537 (cit. on p. 6).

Cui, Hang, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua (2005). “Question answering passage retrieval using dependency relations”. In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 400–407 (cit. on p. 42).

Daiber, Joachim, Max Jakob, Chris Hokamp, and Pablo N Mendes (2013). “Improving efficiency and accuracy in multilingual entity extraction”. In: *Proceedings of the 9th International Conference on Semantic Systems*. ACM (cit. on pp. 134, 142).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (cit. on pp. 51, 52, 58, 65, 72, 80, 91, 94, 98, 156, 163, 166).

Dhingra, Bhuwan, Danish Danish, and Dheeraj Rajagopal (2018). “Simple and Effective Semi-Supervised Question Answering”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 582–587 (cit. on p. 28).

Dhingra, Bhuwan, Hanxiao Liu, Ruslan Salakhutdinov, and William W. Cohen (2017). “A Comparative Study of Word Embeddings for Reading Comprehension”. In: *CoRR* abs/1703.00993. arXiv: `1703.00993` (cit. on p. 139).

Dhingra, Bhuwan, Kathryn Mazaitis, and William W Cohen (2017a). “Quasar: Datasets for question answering by search and reading”. In: *arXiv preprint arXiv:1707.03904* (cit. on pp. 41, 47, 55).

– (2017b). “Quasar: Datasets for Question Answering by Search and Reading”. In: *CoRR* abs/1707.03904. arXiv: `1707.03904` (cit. on p. 87).

Dua, Dheeru, Yizhong Wang, Pradeep Dasigi, et al. (2019). “DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs”. In: *arXiv preprint arXiv:1903.00161* (cit. on pp. 23, 28, 57–59, 163).

Dunn, Matthew, Levent Sagun, Mike Higgins, et al. (2017). “Searchqa: A new q&a dataset augmented with context from a search engine”. In: *arXiv preprint arXiv:1704.05179* (cit. on pp. 29, 55).

Elsken, Thomas, Jan Hendrik Metzen, and Frank Hutter (2018). “Neural architecture search: A survey”. In: *arXiv preprint arXiv:1808.05377* (cit. on p. 8).

Fang, Wei, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li (2016). “Entity Disambiguation by Knowledge and Text Jointly Embedding”. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, pp. 260–269 (cit. on p. 141).

Feng, Minwei, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou (2015). “Applying deep learning to answer selection: A study and an open task”. In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pp. 813–820 (cit. on p. 26).

Ferré, Arnaud (2019). “Représentations vectorielles et apprentissage automatique pour l’alignement d’entités textuelles et de concepts d’ontologie : application à la biologie”. 2019SACLS117. PhD thesis (cit. on p. 165).

Ferret, Olivier, Brigitte Grau, Martine Hurault-Plantet, et al. (2001). “Finding an answer based on the recognition of the question focus”. In: (cit. on p. 39).

Ferrucci, David, Eric Brown, Jennifer Chu-Carroll, et al. (2010). “Building Watson: An overview of the DeepQA project”. In: *AI magazine* 31.3, pp. 59–79 (cit. on p. 55).

Gaizauskas, Robert and Kevin Humphreys (2000). “A combined IR/NLP approach to question answering against large text collections”. In: *Content-Based Multimedia Information Access-Volume 2*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D’INFORMATIQUE DOCUMENTAIRE, pp. 1288–1304 (cit. on p. 41).

Gleize, Martin and Brigitte Grau (2015a). “A Unified Kernel Approach for Learning Typed Sentence Rewritings”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 939–949 (cit. on pp. 42, 43).

– (2015b). “LIMSI-CNRS@ CLEF 2015: Tree Edit Beam Search for Multiple Choice Question Answering.” In: (cit. on p. 46).

Grappy, Arnaud and Brigitte Grau (2010). “Answer type validation in question answering systems”. In: *RIAO 2010, 9th International Conference, Paris, France, April 28-30, 2010, Proceedings*, pp. 9–15 (cit. on p. 119).

Grappy, Arnaud, Brigitte Grau, Mathieu-Henri Falco, et al. (2011). “Selecting answers to questions from Web documents by a robust validation process”. In: *WI* (cit. on pp. 2, 41, 119, 146).

Grau, Brigitte and Anne-Laure Ligozat (2018). “A Corpus for Hybrid Question Answering Systems”. In: *Companion Proceedings of the The Web Conference 2018*. WWW ’18. Lyon, France: International World Wide Web Conferences Steering Committee, pp. 1081–1086 (cit. on p. 38).

Green Jr, Bert F, Alice K Wolf, Carol Chomsky, and Kenneth Laughery (1961). “Baseball: an automatic question-answerer”. In: *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*. ACM, pp. 219–224 (cit. on p. 37).

Habibi, Maryam, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser (2017). “Deep learning with word embeddings improves biomedical named entity recognition”. In: *Bioinformatics* 33.14, pp. i37–i48 (cit. on p. 120).

He, Hua, Kevin Gimpel, and Jimmy Lin (2015). “Multi-perspective sentence similarity modeling with convolutional neural networks”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (cit. on pp. 44, 46).

Heilman, Michael and Noah A Smith (2010). “Tree edit models for recognizing textual entailments, paraphrases, and answers to questions”. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 1011–1019 (cit. on p. 42).

Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, et al. (2015). "Teaching machines to read and comprehend". In: *Advances in neural information processing systems*, pp. 1693–1701 (cit. on pp. 20, 22, 26, 28, 47).

Hermjakob, Ulf (2001). "Parsing and question classification for question answering". In: *Proceedings of the ACL 2001 workshop on open-domain question answering* (cit. on pp. 39, 40).

Hirschman, Lynette, Marc Light, Eric Breck, and John D Burger (1999). "Deep read: A reading comprehension system". In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, pp. 325–332 (cit. on p. 46).

Hosein, Stefan, Daniel Andor, and Ryan McDonald (2019). *Measuring Domain Portability and ErrorPropagation in Biomedical QA*. arXiv: `1909.09704 [cs.CL]` (cit. on pp. 63–65, 92, 99).

Hu, Minghao, Yuxing Peng, Zhen Huang, et al. (2017). "Reinforced mnemonic reader for machine reading comprehension". In: *arXiv preprint arXiv:1705.02798* (cit. on p. 49).

Huang, Hsin-Yuan, Chenguang Zhu, Yelong Shen, and Weizhu Chen (2017). "Fusionnet: Fusing via fully-aware attention with application to machine comprehension". In: *arXiv preprint arXiv:1711.07341* (cit. on pp. 49, 50).

Jia, Robin and Percy Liang (2017). "Adversarial examples for evaluating reading comprehension systems". In: *arXiv preprint arXiv:1707.07328* (cit. on p. 22).

Joshi, Mandar, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer (2017). "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension". In: *arXiv preprint arXiv:1705.03551* (cit. on pp. 29, 55, 87).

Jouis, Christophe, Christophe Jouis, Ismail Biskri, Jean-Gabriel Ganascia, and Magali Roux (2012). "Next Generation Search Engines: Advanced Models for Information Retrieval". In: (cit. on p. 2).

Kamath, Sanjay, Brigitte Grau, and Yue Ma (2017a). "A Study of Word Embeddings for Biomedical Question Answering". In: *SIIM'17* (cit. on p. 112).

– (Nov. 2017b). "A Study of Word Embeddings for Biomedical Question Answering". In: *4e édition du Symposium sur l'Ingénierie de l'Information Médicale*. Toulouse, France (cit. on p. 139).

– (Apr. 2019). "Predicting and Integrating Expected Answer Types into a Simple Recurrent Neural Network Model for Answer Sentence Selection". In: *20th International Conference on Computational Linguistics and Intelligent Text Processing*. La Rochelle, France (cit. on p. 158).

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (cit. on p. 133).

Kobayashi, Tetsuya and Chi-Ren Shyu (2006). "Representing clinical questions by semantic type for better classification". In: *AMIA Annual Symposium Proceedings*, pp. 987–987 (cit. on p. 119).

Kolomiyets, Oleksandr and Marie-Francine Moens (2011). "A survey on question answering technology from an information retrieval perspective". In: *Information Sciences* 181.24 (cit. on p. 119).

Kwiatkowski, Tom, Jennimaria Palomaki, Olivia Rhinehart, et al. (2019). "Natural questions: a benchmark for question answering research". In: (cit. on pp. 26, 29, 59, 65, 92, 99).

Lai, Tuan Manh, Trung Bui, and Sheng Li (2018). "A review on deep learning techniques applied to answer selection". In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2132–2144 (cit. on pp. 25, 44, 45).

Lavergne, Thomas, Olivier Cappé, and François Yvon (July 2010). "Practical Very Large Scale CRFs". In: *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. Uppsala, Sweden: Association for Computational Linguistics, pp. 504–513 (cit. on pp. 39, 149).

Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, et al. (2019). "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *CoRR* abs/1901.08746. arXiv: 1901.08746 (cit. on pp. 63, 90–93, 96–99, 163).

Lee, Kenton, Ming-Wei Chang, and Kristina Toutanova (2019). "Latent Retrieval for Weakly Supervised Open Domain Question Answering". In: *arXiv preprint arXiv:1906.00300* (cit. on pp. 54, 55, 155, 156, 160, 166).

Levy, Omer, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer (2017). "Zero-Shot Relation Extraction via Reading Comprehension". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342 (cit. on p. 58).

Lewis, Patrick, Ludovic Denoyer, and Sebastian Riedel (2019). "Unsupervised Question Answering by Cloze Translation". In: *arXiv preprint arXiv:1906.04980* (cit. on p. 1).

Li, Xin and Dan Roth (2002). "Learning question classifiers". In: *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pp. 1–7 (cit. on p. 40).

Lin, Yankai, Haozhe Ji, Zhiyuan Liu, and Maosong Sun (2018). "Denoising distantly supervised open-domain question answering". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1736–1745 (cit. on pp. 29, 55, 73, 74, 76, 91, 101, 131, 155).

Liu, Yinhan, Myle Ott, Naman Goyal, et al. (2019). "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (cit. on p. 97).

Luo, Zhihui, Meliha Yetisgen-Yildiz, and Chunhua Weng (2011). "Dynamic categorization of clinical research eligibility criteria by hierarchical clustering". In: *Journal of biomedical informatics* 44.6, pp. 927–935 (cit. on p. 119).

Madabushi, Harish Tayyar and Mark Lee (2016). "High accuracy rule-based question classification using question syntax and semantics". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (cit. on pp. 125, 127, 129, 133–135, 149, 164).

Mikolov, T, K Chen, G Corrado, and J Dean (2013). "Efficient estimation of word representations in vector space (2013). arXiv preprint". In: *arXiv preprint arXiv:1301.3781*, pp. 1532–1543 (cit. on pp. 85, 86).

Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin (2018). “Advances in Pre-Training Distributed Word Representations”. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (cit. on p. 139).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*, pp. 3111–3119 (cit. on pp. 121, 133, 139, 141).

Miller, George A (1995). “WordNet: a lexical database for English”. In: *Communications of the ACM* 38.11, pp. 39–41 (cit. on p. 40).

Moldovan, Dan, Sanda Harabagiu, Marius Pasca, et al. (1999). “Lasso: A tool for surfing the answer net”. In: (cit. on pp. 39, 40).

Moraes, Rodrigo, JoãO Francisco Valiati, and Wilson P GaviãO Neto (2013). “Document-level sentiment classification: An empirical comparison between SVM and ANN”. In: *Expert Systems with Applications* 40.2, pp. 621–633 (cit. on p. 6).

Moreno, Jose G, Romaric Besançon, Romain Beaumont, et al. (2017). “Combining word and entity embeddings for entity linking”. In: *European Semantic Web Conference*. Springer, pp. 337–352 (cit. on p. 141).

Neves, Mariana and Milena Kraus (2016). “BioMedLAT corpus: Annotation of the lexical answer type for biomedical questions”. In: *Proceedings of the Open Knowledge Base and Question Answering Workshop (OKBQA 2016)*, pp. 49–58 (cit. on pp. 121, 165).

Paris, Cecile L (1985). “Towards more graceful interaction: a survey of question-answering programs”. In: (cit. on p. 37).

Peñas, Anselmo, Eduard Hovy, Pamela Forner, et al. (2013). “QA4MRE 2011-2013: Overview of question answering for machine reading evaluation”. In: *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, pp. 303–320 (cit. on pp. 20, 46).

Peng, Fuchun, Ralph Weischedel, Ana Licuanan, and Jinxi Xu (2005). “Combining deep linguistics analysis and surface pattern learning: A hybrid approach to Chinese definitional question answering”. In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 307–314 (cit. on p. 42).

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (cit. on pp. 70, 133, 139).

Peters, Matthew E, Waleed Ammar, Chandra Bhagavatula, and Russell Power (2017). “Semi-supervised sequence tagging with bidirectional language models”. In: *arXiv preprint arXiv:1705.00108* (cit. on pp. 51, 80).

Peters, Matthew, Mark Neumann, Mohit Iyyer, et al. (2018a). “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237 (cit. on pp. 51, 80).

– (2018b). “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Vol. 1 (cit. on p. 72).

Petrova, Alina, Yue Ma, George Tsatsaronis, et al. (2015). “Formalizing biomedical concepts from textual definitions”. In: *J. Biomedical Semantics* 6, p. 22 (cit. on p. 119).

Poon, Hoifung, Janara Christensen, Pedro Domingos, et al. (2010). “Machine reading at the university of washington”. In: *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*. Association for Computational Linguistics, pp. 87–95 (cit. on p. 46).

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih (2004). *Natural language inference via dependency tree mapping: An application to question answering*. Tech. rep. (cit. on p. 42).

Radev, Dragomir, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal (2002). “Probabilistic question answering on the web”. In: *Proceedings of the 11th international conference on World Wide Web*. ACM, pp. 408–419 (cit. on p. 39).

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). “Improving language understanding by generative pre-training”. In: (cit. on p. 51).

Rajpurkar, Pranav, Robin Jia, and Percy Liang (2018). “Know What You Don’t Know: Unanswerable Questions for SQuAD”. In: *arXiv preprint arXiv:1806.03822* (cit. on pp. 23, 28, 53, 57, 59, 101, 163).

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang (2016). “SQuAD: 100,000+ Questions for Machine Comprehension of Text”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392 (cit. on pp. 11, 21, 26, 32, 33, 47, 48, 59, 64, 65, 69, 112, 124, 134, 145, 154, 163).

Rao, Jinfeng, Hua He, and Jimmy Lin (2016). “Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM ’16 (cit. on pp. 44, 134, 136, 137).

Ravichandran, Deepak and Eduard Hovy (2002). “Learning surface text patterns for a question answering system”. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 41–47 (cit. on p. 42).

Riloff, Ellen and Michael Thelen (2000). “A Rule-based Question Answering System for Reading Comprehension Tests”. In: *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems* (cit. on p. 46).

Russakovsky, Olga, Jia Deng, Hao Su, et al. (Dec. 2015). “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3, pp. 211–252 (cit. on p. 80).

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf (2019). “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108* (cit. on p. 97).

Santos, Cicero dos, Ming Tan, Bing Xiang, and Bowen Zhou (2016). “Attentive pooling networks”. In: *arXiv preprint arXiv:1602.03609* (cit. on p. 45).

Schulze, Frederik, Ricarda Schüler, Tim Draeger, et al. (2016). “Hpi question answering system in bioasq 2016”. In: *Proceedings of the Fourth BioASQ workshop*, pp. 38–44 (cit. on p. 62).

Seo, Minjoon, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi (2016). “Bidirectional attention flow for machine comprehension”. In: *arXiv preprint arXiv:1611.01603* (cit. on pp. 11, 49, 71, 77, 78).

Severyn, Aliaksei and Alessandro Moschitti (2013). “Automatic feature engineering for answer selection and extraction”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 458–467 (cit. on pp. 42, 157, 158).

– (2015). “Learning to rank short text pairs with convolutional deep neural networks”. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM (cit. on p. 44).

Shen, Yelong, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen (2017). “Reasonet: Learning to stop reading in machine comprehension”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1047–1055 (cit. on p. 49).

Sherkat, Ehsan and Evangelos E Milios (2017). “Vector embedding of wikipedia concepts and entities”. In: *International Conference on Applications of Natural Language to Information Systems*. Springer, pp. 418–428 (cit. on p. 139).

Shih, Cheng-Wei, Min-Yuh Day, Tzong-Han Tsai, et al. (2005). “ASQA: Academia sinica question answering system for NTCIR-5 CLQA”. In: (cit. on p. 42).

Singh, Amit (2012). “Entity based q&a retrieval”. In: *Proceedings of the 2012 Joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, pp. 1266–1277 (cit. on p. 54).

Socher, Richard, John Bauer, Christopher D Manning, et al. (2013). “Parsing with compositional vector grammars”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 455–465 (cit. on p. 6).

Stenetorp, Pontus, Sampo Pyysalo, Goran Topić, et al. (2012). “BRAT: a web-based tool for NLP-assisted text annotation”. In: *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 102–107 (cit. on p. 108).

Stoyanchev, Svetlana, Young Chol Song, and William Lahti (2008). “Exact phrases in information retrieval for question answering”. In: *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*. Association for Computational Linguistics, pp. 9–16 (cit. on p. 41).

Talmor, Alon and Jonathan Berant (July 2019). “MultiQA: An Empirical Investigation of Generalization and Transfer in Reading Comprehension”. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4911–4921 (cit. on pp. 94, 96, 156, 165).

Tan, Ming, Cicero dos Santos, Bing Xiang, and Bowen Zhou (2015). “Lstm-based deep learning models for non-factoid answer selection”. In: *arXiv preprint arXiv:1511.04108* (cit. on p. 45).

Tayyar Madabushi, Harish, Mark Lee, and John Barnden (2018). “Integrating Question Classification and Deep Learning for improved Answer Selection”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics (cit. on pp. 46, 128, 132–136, 138).

Tsatsaronis, George, Georgios Balikas, Prodromos Malakasiotis, et al. (Apr. 2015). “An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition”. In: *BMC Bioinformatics* 16.1, p. 138 (cit. on pp. 30, 61, 64, 120, 146).

Tymoshenko, Kateryna and Alessandro Moschitti (2018). “Cross-Pair Text Representations for Answer Sentence Selection”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (cit. on p. 136).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, et al. (2017). “Attention is all you need”. In: *Advances in neural information processing systems*, pp. 5998–6008 (cit. on pp. 50, 51, 72).

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly (2015). “Pointer networks”. In: *Advances in Neural Information Processing Systems*, pp. 2692–2700 (cit. on p. 49).

Voorhees, Ellen M (2001). “The TREC question answering track”. In: *Natural Language Engineering* 7.4, pp. 361–378 (cit. on p. 29).

– (2002). “The evaluation of question answering systems: Lessons learned from the TREC QA track”. In: *Question Answering: Strategy and Resources Workshop Program*, p. 6 (cit. on p. 60).

Wang, Mengqiu and Christopher D Manning (2010). “Probabilistic tree-edit models with structured latent variables for textual entailment and question answering”. In: *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pp. 1164–1172 (cit. on p. 42).

Wang, Mengqiu, Noah A Smith, and Teruko Mitamura (2007). “What is the Jeopardy model? A quasi-synchronous grammar for QA”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 22–32 (cit. on pp. 19, 25, 43, 60, 157).

Wang, Shuohang and Jing Jiang (2016). “Machine comprehension using match-lstm and answer pointer”. In: *arXiv preprint arXiv:1608.07905* (cit. on p. 49).

Wang, Shuohang, Mo Yu, Xiaoxiao Guo, et al. (2018). “R 3: Reinforced ranker-reader for open-domain question answering”. In: *Thirty-Second AAAI Conference on Artificial Intelligence* (cit. on pp. 29, 55).

Wang, Shuohang, Mo Yu, Jing Jiang, et al. (2017). “Evidence aggregation for answer re-ranking in open-domain question answering”. In: *arXiv preprint arXiv:1711.05116* (cit. on p. 55).

Weissenborn, Dirk, Georg Wiese, and Laura Seiffe (2017). “Making neural qa as simple as possible but not simpler”. In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 271–280 (cit. on pp. 49, 119).

Wiese, Georg, Dirk Weissenborn, and Mariana Neves (Aug. 2017a). “Neural Domain Adaptation for Biomedical Question Answering”. In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 281–289 (cit. on pp. 7, 93).

– (2017b). “Neural Domain Adaptation for Biomedical Question Answering”. In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 281–289 (cit. on pp. 64, 81, 96, 113, 114).

Wiese, Georg, Dirk Weissenborn, and Mariana Neves (Aug. 2017c). "Neural Question Answering at BioASQ 5B". In: *BioNLP 2017*. Vancouver, Canada, Association for Computational Linguistics, pp. 76–79 (cit. on pp. 62, 106, 111, 113–115, 120).

Wolf, Thomas, Lysandre Debut, Victor Sanh, et al. (2019). *Transformers: State-of-the-art Natural Language Processing*. arXiv: `1910.03771 [cs.CL]` (cit. on p. 73).

Woods, William A (1973). "Progress in natural language understanding: an application to lunar geology". In: *Proceedings of the June 4-8, 1973, national computer conference and exposition*. ACM, pp. 441–450 (cit. on p. 37).

Xiong, Caiming, Victor Zhong, and Richard Socher (2016). "Dynamic coattention networks for question answering". In: *arXiv preprint arXiv:1611.01604* (cit. on p. 49).

Yamada, Ikuya, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji (2016). "Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, pp. 250–259 (cit. on p. 141).

Yang, Yi, Wen-tau Yih, and Christopher Meek (2015). "Wikiqa: A challenge dataset for open-domain question answering". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2013–2018 (cit. on pp. 25, 43, 134).

Yang, Zhilin, Zihang Dai, Yiming Yang, et al. (2019). "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *arXiv preprint arXiv:1906.08237* (cit. on p. 97).

Yang, Zhilin, Peng Qi, Saizheng Zhang, et al. (2018). "Hotpotqa: A dataset for diverse, explainable multi-hop question answering". In: *arXiv preprint arXiv:1809.09600* (cit. on pp. 22, 23, 28, 58, 59, 163).

Yang, Zi, Yue Zhou, and Eric Nyberg (2016). "Learning to answer biomedical questions: Oaqa at bioasq 4b". In: *Proceedings of the Fourth BioASQ workshop*, pp. 23–37 (cit. on pp. 62, 119).

Yao, Xuchen, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark (June 2013a). "Answer Extraction as Sequence Tagging with Tree Edit Distance". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 858–867 (cit. on pp. 157, 158).

– (2013b). "Answer extraction as sequence tagging with tree edit distance". In: *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 858–867 (cit. on p. 42).

Yih, Wen-tau, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak (2013). "Question answering using enhanced lexical semantic models". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1744–1753 (cit. on p. 42).

Yin, Wenpeng, Hinrich Schütze, Bing Xiang, and Bowen Zhou (2016). "Abcnn: Attention-based convolutional neural network for modeling sentence pairs". In: *Transactions of the Association for Computational Linguistics* 4 (cit. on p. 44).

Yoon, Wonjin, Jinhyuk Lee, Donghyeon Kim, Minbyul Jeong, and Jaewoo Kang (2019). *Pre-trained Language Model for Biomedical Question Answering*. arXiv: `1909.08229 [cs.CL]` (cit. on pp. 64, 65, 91, 96, 97, 101).

Yu, Adams Wei, David Dohan, Minh-Thang Luong, et al. (2018). “Qanet: Combining local convolution with global self-attention for reading comprehension”. In: *arXiv preprint arXiv:1804.09541* (cit. on pp. 50, 72).

Yu, Lei, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman (2014). “Deep learning for answer sentence selection”. In: *arXiv preprint arXiv:1412.1632* (cit. on p. 44).

Zhang, Dell and Wee Sun Lee (2003). “Question classification using support vector machines”. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, pp. 26–32 (cit. on p. 40).

List of Figures

List of Tables

Ecole doctorale n°580
Sciences et technologies de l'information et de la communication (STIC)

Titre : Question-réponse utilisant des données et modèles hybrides

Mots clés : Question-réponse, Traitement langage naturel, Apprentissage automatique

Résumé : La recherche de réponses à des questions relève de deux disciplines : le traitement du langage naturel et la recherche d'information. L'émergence de l'apprentissage profond dans plusieurs domaines de recherche tels que la vision par ordinateur, le traitement du langage naturel etc. a conduit à l'émergence de modèles de bout en bout. Les travaux actuels de l'état de l'art en question-réponse (QR) visent à mettre en œuvre de tels modèles.

Dans le cadre du projet GoASQ, l'objectif est d'étudier, comparer et combiner différentes approches pour répondre à des questions formulées en langage naturel sur des données textuelles, en domaine ouvert et en domaine biomédical. Ce travail se concentre principalement sur 1) la construction de modèles permettant de traiter des ensembles de données à petite et à grande échelle, et 2) l'exploitation de connaissances sémantiques pour répondre aux questions par leur intégration dans les différents modèles. Nous visons à fusionner des connaissances issues de textes libres, d'ontologies, de représentations d'entités, etc.

Afin de faciliter l'utilisation des modèles neuronaux sur des données de domaine de spécialité, nous nous plaçons dans le cadre de l'adaptation de domaine. Nous avons proposé deux modèles de tâches de QR différents, évalués sur la tâche BIOASQ de réponse à des questions biomédicales. Nous montrons par nos résultats expérimentaux que le modèle de QR ouvert convient mieux qu'une modélisation de type Compréhension machine. Nous pré-entrainons le modèle de Compréhension machine, qui sert de base à notre modèle, sur différents ensembles de données pour montrer la variabilité des performances. Nous constatons que l'utilisation d'un ensemble de données particulier pour le pré-entraînement donne les meilleurs résultats lors du test et qu'une combinaison de quatre jeux de données donne les meilleurs résultats lors de l'adaptation au domaine biomédical. Nous avons testé des modèles de langage à grande échelle, comme BERT, qui sont adaptés à la tâche de réponse aux questions. Les performances varient en fonction du type des données utilisées pour pré-entrainer BERT. Ainsi, le modèle de langue appris sur des données biomédicales, BIOBERT, constitue le meilleur choix pour le QR biomédical.

Les modèles d'apprentissage profond visent à fonctionner de bout en bout. Les informations sémantiques provenant de sources de connaissances construites par des experts n'y sont généralement pas introduites. Nous avons annoté manuellement et automatiquement un jeu de données par les variantes des réponses de BIOASQ et montré l'importance d'apprendre un modèle de QR avec ces variantes. Ces types sont ensuite utilisés pour mettre en évidence les entités dans les jeux de données, ce qui montre des améliorations sur l'état de l'art. Par ailleurs l'exploitation de représentations vectorielles d'entités dans les modèles se montre positif pour le domaine ouvert.

Nous faisons l'hypothèse que les résultats obtenus à partir de modèles d'apprentissage profond peuvent être encore améliorés en utilisant des traits sémantiques et des traits collectifs calculés à partir des différents paragraphes sélectionnés pour répondre à une question. Nous utilisons des modèles de classification binaires pour améliorer la prédiction de la réponse parmi les K candidats à l'aide de ces caractéristiques, conduisant à un modèle hybride qui surpasse les résultats de l'état de l'art. Enfin, nous avons évalué des modèles de QR ouvert sur des ensembles de données construits pour les tâches de Compréhension machine et Sélection de phrases. Nous montrons la différence de performance lorsque la tâche à résoudre est une tâche de QR ouverte et soulignons le fossé important qu'il reste à franchir dans la construction de modèles de bout en bout pour la tâche complète de réponse aux questions.

Title : Question Answering with Hybrid Data and Models

Keywords : Question Answering, Natural Language Processing, Machine Learning

Abstract : Question Answering is a discipline which lies in between natural language processing and information retrieval domains. Emergence of deep learning approaches in several fields of research such as computer vision, natural language processing, speech recognition etc. has led to the rise of end-to-end models.

In the context of GoASQ project we investigate, compare and combine different approaches for answering questions formulated in natural language over textual data on open domain and biomedical domain data. The thesis work mainly focuses on 1) Building models for small scale and large scale datasets, and 2) Leveraging structured and semantic information into question answering models. Hybrid data in our research context is fusion of knowledge from free text, ontologies, entity information etc. applied towards free text question answering.

The current state-of-the-art models for question answering use deep learning based models. In order to facilitate using them on small scale datasets on closed domain data, we propose to use domain adaptation. We model the BIOASQ biomedical question answering task dataset into two different QA task models and show how the *Open Domain Question Answering* task suits better than the *Reading Comprehension* task by comparing experimental results. We pre-train the *Reading Comprehension* model with different datasets to show the variability in performance when these models are adapted to biomedical domain. We find that using one particular dataset (SQUAD v2.0 dataset) for pre-training performs the best on single dataset pre-training and a combination of four Reading Comprehension datasets performed the best towards the biomedical domain adaptation.

We perform some of the above experiments using large scale pre-trained language models like BERT which are fine-tuned to the question answering task. The performance varies based on the type of data used to pre-train BERT. For BERT pre-training on the language modelling task, we find the biomedical data trained BIOBERT to be the best choice for biomedical QA. Since deep learning models tend to function in an end-to-end fashion, semantic and structured information coming from expert annotated information sources are not explicitly used. We highlight the necessity for using *Lexical and Expected Answer Types* in open domain and biomedical domain question answering by performing several verification experiments. These types are used to highlight entities in two QA tasks which shows improvements while using entity embeddings based on the answer type annotations. We manually annotated an answer variant dataset for BIOASQ and show the importance of learning a QA model with answer variants present in the paragraphs.

Our hypothesis is that the results obtained from deep learning models can further be improved using semantic features and collective features from different paragraphs for a question. We propose to use ranking models based on binary classification methods to better rank Top-1 prediction among Top-K predictions using these features, leading to an hybrid model that outperforms state-of-art-results on several datasets. We experiment with several overall Open Domain Question Answering models on QA sub-task datasets built for *Reading Comprehension* and *Answer Sentence Selection* tasks. We show the difference in performance when these are modelled as overall QA task and highlight the wide gap in building end-to-end models for overall question answering task.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France