# Deep Neural Networks for Biomedical Question Answering


**Dimitris Pappas**

Department of Informatics
Athens University of Economics and Business

**Supervisors:** Ion Androutsopoulos
Haris Papageorgiou

May 2023

I would like to dedicate this thesis to my parents and my wife for their unconditional support

My unlimited respect and gratitude to Ion, Makis and the NLP group of AUEB

# Abstract

In this thesis, we advance biomedical Question Answering (QA). The first part of the thesis focuses on retrieving abstracts of scientific literature given a natural language question submitted by a biomedical expert. We propose multiple state-of-the-art deep-learning models for biomedical document retrieval and snippet extraction. Joint models that simultaneously retrieve documents and snippets are also proposed, which improve the results even further. The best deep learning model was made publicly available as a research prototype during the Coronavirus pandemic to aid researchers around the world. In collaboration with biomedical experts, we also deployed one of our deep learning models for document retrieval and developed a literature identification system for systematic reviews.

One major issue in biomedical QA is the scarcity of human-annotated data as annotation of biomedical literature demands human expertise and time. Therefore in the second part of the thesis, we develop two new artificial datasets for biomedical cloze-style QA and make them publicly available. We followed a methodology previously used in news articles and books and extracted millions of artificial training examples that can be used to train data-demanding deep learning models. Through human performance evaluation, we show that human experts outperform non-experts in the resulting cloze-style QA task, which supports the claim that human expertise is essential for biomedical QA. We developed and trained new deep-learning models for reading comprehension using our new datasets. Our models outperform previously proposed deep learning models for cloze-style QA, as well as four strong baselines. In experiments conducted on a sample of the dataset, the best model outperformed all human non-experts and achieved competitive results compared to biomedical experts.

In the third part of the thesis, we train deep learning models for factoid QA in two well-established biomedical datasets. Given a snippet of text and a question, a span of the snippet is selected as an answer. We examine six techniques for offline data augmentation (data augmentation applied before training). We show that in biomedical factoid QA, all data augmentation techniques improve performance, even when fine-tuning very large pre-trained language models. We also show that using one of the artificial datasets created in this thesis acts as a good data augmentation technique.

# Table of contents

# List of figures

# List of tables

# List of Thesis Publications

## Relevant to Chapter 4

- *Brokos, G., Liosis, P., McDonald, R., Pappas, D., and Androutsopoulos, I. (2018a). AUEB at BioASQ 6: Document and Snippet Retrieval. In Proceedings of the 6th BioASQ Workshop, pages 30–39, Brussels, Belgium. [24]*

- *Pappas, D., McDonald, R., Brokos, G.-I., and Androutsopoulos, I. (2019). AUEB at BioASQ 7: document and snippet retrieval. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 607–623, Wurzburg, Germany. [143]*

- *Pappas, D., Stavropoulos, P., and Androutsopoulos, I. (2020a). AUEB-NLP at BioASQ 8: Biomedical Document and Snippet Retrieval. In Proceedings of the 8th BioASQ workshop at the Conference and Labs of the Evaluation Forum (CLEF 2020), pages 41–54, Thessaloniki, Greece. [144]*

- *Pappas, D. and Androutsopoulos, I. (2021). A neural model for joint document and snippet ranking in question answering for large document collections. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th nternational Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3896–3907, Online. Association for Computational Linguistics. [140]*

- *Adam, G. P., Pappas, D., Papageorgiou, H., Evangelou, E., and Trikalinos, T. A. (2022b). Title of project: A novel tool that allows interactive screening of pubmed citations showed promise for the semi-automation of identification of biomedical literature. Journal of Clinical Epidemiology. [3]*

# Relevant to Chapter 5

- *Pappas, D., Androutsopoulos, I., and Papageorgiou, H. (2018a). BioRead: A new dataset for biomedical reading comprehension. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan. European Language Resources Association (ELRA).* [141]

- *Pappas, D., Stavropoulos, P., Androutsopoulos, I., and McDonald, R. (2020b). BioMRC: A dataset for biomedical machine reading comprehension. In Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing, pages 140–149, Online. Association for Computational Linguistics.* [145]

# Relevant to Chapter 6

- *Pappas, D., Malakasiotis, P., and Androutsopoulos, I. (2022). Data augmentation for biomedical factoid question answering. In Proceedings of the 21st Workshop on Biomedical Language Processing, pages 63–81, Dublin, Ireland. Association for Computational Linguistics.* [142]

# List of Thesis Resources

## Relevant to Chapter 4

### Data

- We released instructions and code to modify the Natural Questions dataset to a version useful for Information Retrieval that contains $110,589$ questions and $2,684,631$ paragraphs.
  Link: `https://github.com/dpappas/Joint_Document_Snippet_Ranking`

### Pre-trained Embeddings

- 30-dimensional English biomedical word embeddings trained using WORD2VEC with more than $28M$ abstracts of biomedical articles.
  Link: `https://archive.org/details/pubmed2018_w2v_30D`
- Biomedical co-occurrence graph embeddings trained using graph convolutional networks (see more in section 4.5).
  Link: `https://archive.org/details/pubtator_word_embeddings`

### Code

- Code to replicate experiments and jointly train document retrieval and sentence retrieval models related to Pappas et al.[144, 140]. Link: `https://github.com/dpappas/Joint_Document_Snippet_Ranking`
- Code to train retrieval models and pipelines for document retrieval and sentence retrieval models related to Pappas et al.[143].
  Link: `https://github.com/dpappas/pytorch_pacrr_and_posit_drmm`

### Other

- A search engine for Covid-19-related biomedical articles (see more in section 4.9)
  Link: https://cslab241.cs.aueb.gr:5000/
- News article on greek media (page 6).
  Link: https://www.yumpu.com/en/document/view/62288498/-49

## Relevant to Chapter 5

### Data

- BioRead is a new publicly available cloze-style biomedical machine reading comprehension (MRC) dataset with approximately 16.4 million passage-question instances (see more in section 5.3).
  Link: https://archive.org/details/bioread_dataset.tar
- BIOMRC is a large-scale cloze-style biomedical MRC dataset with more than 812$K$ instances. Care was taken to reduce noise, compared to the previous BioRead dataset. We make the new dataset available in three different sizes (see more in section 5.3).
  Link: https://huggingface.co/datasets/viewer/?dataset=biomrc&config=biomrc_large_A

### Code

- We release the code to replicate experiments on BioRead data.
  Link: https://github.com/dpappas/BIOREAD_code.
- We release the code to replicate experiments on BIOMRC data.
  Link: https://github.com/PetrosStav/BioMRC_code.

### Other

- We developed and release an HTML template for a leaderboard for comparison of MRC models. The template is tailored for BioRead and can be modified for the BIOMRC dataset.
  Link: https://dvpappas89.wixsite.com/bioread2.

# Relevant to Chapter 6

## Code

- We release the code for data augmentation and the code to replicate the experiments mentioned in Pappas et al. [142].
  Link: `https://github.com/dpappas/Data-Augmentation-for-Biomedical-Factoid-Question-Answering`.

# Relevant to Appendix B (Dense retrieval)

- We release the code for our dense retrieval models.
  Link: `https://github.com/dpappas/adhoc_retrieval`.

# Disclaimer

This thesis explores the use of deep learning models for question-answering and text retrieval in the medical domain. It is important to note that while the models have the potential to improve the efficiency and accuracy of biomedical research, they may also raise ethical concerns. Specifically, there is a risk that relying solely on machine-generated answers could lead to errors or biases in the decision-making process.

The author acknowledges the importance of using the models to supplement the expertise of biomedical experts and not replace it. While the models have been trained on scientific publications they are still fallible and should always be used with caution.

It is crucial that any results generated by the models are cross-checked by human experts to ensure their accuracy and reliability. Additionally, the ethical implications of relying on machine-generated answers should be carefully considered and weighed against the potential benefits.

The author recognizes the potential limitations of the models and is committed to ensuring that they are used ethically and responsibly in the pursuit of advancing biomedical research.

# Chapter 1

# Overview of the thesis

## 1.1 Question Answering

Question answering (QA) is a sub-field of natural language processing (NLP) focused on developing algorithms that can automatically answer questions posed in natural language. QA has a long history in NLP [173] and it has become increasingly important as the amount of digital information is constantly increasing. With the advancements in deep learning models, Question Answering systems have become more sophisticated and accurate in understanding the intent of a question and providing the most relevant answer [1, 126, 220]. QA systems are designed to automatically process the abundance of data and answer questions posed in natural language. As advancements are made in NLP research, question answering has been found to be beneficial in a range of tasks such as information retrieval, machine translation, and dialog systems.

There are several types of question-answering systems, each with its own unique characteristics and capabilities. *Rule-based systems* rely on a set of predefined patterns or rules that match the question to the appropriate answer in a text. They are typically based on simple algorithms and are limited in their ability to understand the underlying meaning of a question. Rule-based systems fail to understand questions that are not in the predefined patterns or rules, and they may not be able to handle synonyms or paraphrasing.

*Retrieval-based* QA systems [89, 225] use information retrieval techniques to retrieve a set of relevant documents in response to a question and then extract the answer from those documents. A retrieval-based QA system typically employs information retrieval techniques such as keyword search or semantic search. Once the system has found a set of documents that are relevant to the question, it then needs to rank them in order to find the most likely answer. This can be done using a variety of methods such as TF-IDF or BM25 ranking algorithms.

Snippet extraction is typically used in information retrieval-based question answering systems, where the system retrieves relevant text passages from a large text corpus, and then uses the retrieved text passages to answer the question. The goal of snippet extraction is to retrieve the most relevant text spans, also known as snippets, that can be used to answer the question.

*Interactive Question Answering (IQA) systems* are typically used in information retrieval and search systems. They allow users to interactively refine their queries and receive feedback on the quality of their query results using various tools such as query suggestion, query expansion, and query reformulation.

QA systems for *structured data* [90, 111, 72] are a type of QA system that utilize pre-organized and well-defined data, usually stored in a relational database or a knowledge graph, to answer questions. These systems use structured data, such as tables or knowledge graphs, to extract the relevant information to the question and provide the answer.

*Conversational agents* [216] are designed to interact with users in a natural and human-like way. They use NLP techniques to understand the user's questions and provide appropriate answers. They can engage in a back-and-forth dialogue with the user and can keep track of the conversation context to provide more accurate answers.

*Generative question answering* [97] is a type of QA system that generates answers to questions, rather than extracting them from a pre-existing text or database. These systems use generative models, such as neural networks, to generate the answer to a question. They are able to understand the underlying meaning of a question, generate a coherent and informative answer, and sometimes even generate an answer for questions that have no answer in the text. They can also generate answers to follow-up questions, providing a more natural conversation-like experience.

Advances in all these types of QA systems have led to the development of many different applications for question-answering systems such as information retrieval systems, chatbots, digital assistants, or even for educational purposes such as quiz generators. QA systems can be integrated as a component in larger chatbot systems to handle natural language understanding and answer generation tasks. This is especially useful in cases where the chatbot is designed to handle a specific domain or task, such as customer service [228, 36] or healthcare[199, 146, 4], where the QA system can be fine-tuned to that domain or task. However, it's also possible to have a QA system as a standalone chatbot. In this case, the QA system would be responsible for handling the entire conversation, including understanding the user's question, generating an answer, and handling follow-up questions[38, 22]. This can also make digital assistants more interactive and user-friendly. Finally, they can be used as part of educational applications to generate quizzes, test questions, or interactive

activities[185], by using natural language understanding to interpret the quiz topic and generate questions that are relevant to the topic.

The purpose of this thesis is to improve already existing deep learning methods for biomedical question answering, information retrieval and machine reading comprehension in the scientific literature from the biomedical domain, by exploring various neural architectures and training techniques. The aim is to achieve a higher accuracy and better generalization in biomedical QA. Additionally, we will also examine if jointly trained neural networks for snippet retrieval are better than pipeline approaches. By jointly training, we mean training the model end-to-end rather than training different components of the QA system separately and then combining them. This approach has been shown to be beneficial in other NLP tasks and we want to investigate if it's also the case for biomedical QA.

Furthermore, a significant part of this thesis is devoted to creating new large-scale biomedical machine reading comprehension (MRC) datasets. Machine Reading Comprehension (MRC) is a form of question answering that aims to develop algorithms and models that can understand and answer questions about text data in a way that mimics human reading comprehension. MRC systems take an input text and a question, and they try to identify the answer to the question within the text. The task is to identify the exact span of the text which contains the answer. By creating new biomedical MRC datasets, we aim to increase the availability of data for training and evaluating biomedical MRC models, which will ultimately lead to more accurate and robust QA systems. We hope to encourage further research in the field of biomedical QA and machine reading comprehension.

There are a few key reasons why the biomedical domain is much different than generic domains. The biomedical domain contains a vast amount of specialized terminology (jargon), vocabularies, and ontologies that can be difficult for non-experts to understand. The domain is constantly evolving as new discoveries are made and new treatments are developed, which can make it difficult to keep up with the latest advances.

As an example one could consider the term 'radiation therapy' which is used to describe both high-energy radiation for ttherapy and low-energy radiation for diagnostic purposes. High-energy radiation, also known as ionizing radiation, has enough energy to remove tightly bound electrons from atoms, which can damage or destroy living cells. Examples of high-energy radiation include X-rays and gamma rays. These types of radiation are used in diagnostic imaging such as X-rays, CT scans, and PET scans, as well as in radiation therapy for cancer treatment. On the other hand, low-energy radiation, also known as non-ionizing radiation, does not have enough energy to remove electrons from atoms. Examples of low-energy radiation include radio waves, microwaves, and infrared radiation. These types of radiation are used in various diagnostic procedures such as MRI and ultrasound imaging.

They do not have the same potential to damage living cells as high-energy radiation, but prolonged exposure may still pose health risks. By utilizing their background knowledge and analyzing the surrounding context, a biomedical expert is able to understand the meaning and implications of the phrase 'radiation therapy' when encountered in a biomedical text.

When applying biomedical NLP experts are needed because they can provide domain-specific knowledge and understanding that is essential for accurately processing and interpreting the biomedical text data. Biomedical NLP tasks often involve complex medical concepts, terminologies, and relations that may not be well understood by non-experts.

For example, biomedical experts can help identify and resolve ambiguities in the text data, such as different meanings of the same medical term, synonyms, and acronyms. They can also help to design and evaluate biomedical NLP models by providing feedback on the performance of the models in terms of their ability to understand and extract relevant information from the text data.

Additionally, biomedical experts can assist in the annotation and curation of the data, which is an important step in the development of biomedical NLP systems. They can help to identify the relevant entities and relations in the text, and provide expert knowledge to guide the annotation process. They can also help to ensure the quality and consistency of the annotation.

## 1.2 Question Answering in the Biomedical Domain

Question Answering is yet an unsolved problem of Natural Language Processing. The open-access repositories of biomedical documents have millions of entries and are yearly increasing in size adding more documents to their databases. The vocabulary of biomedical documents is constantly increasing as new biomedical terms are introduced. The biomedical ontologies (like Mesh[1] or the disease ontology[2]) are as well increasing in size adding more and more biomedical named entities. The most important differences of the biomedical NLP domain compared to generic domain NLP are the following:

#### Need of Human Expertise

Annotating biomedical texts is a crucial task that requires a high level of expertise, as it involves identifying and labeling various elements in a text such as named entities, relations, and events. The process of annotation is done manually by experts in the biomedical field, who are able to understand the context and meaning of the text, as well as identify the

[1] `https://www.nlm.nih.gov/mesh/`
[2] `http://disease-ontology.org/`

relevant information. The importance of accurate annotation is paramount as it serves as the foundation for many biomedical NLP tasks such as information retrieval, question answering, and summarization.

### Complex Technical Vocabulary and Abbreviations

There are multiple entities often abbreviated[3] and traditional language models trained on generic domain data often fail in the biomedical domain without further fine-tuning. This is further discussed in the paper of Tinn et al. [181] where they present the limitations of traditional language models trained on generic domain data in the biomedical domain, and it highlights the importance of fine-tuning these models on biomedical data to improve their performance on biomedical NLP tasks.

### Universal Annotation of Biomedical Entities

Many different biomedical Named Entity Recognition (NER) corpora contain varying annotations due to their diverse nature and the different types of entities and relations they aim to capture. These variations in annotations can be attributed to factors such as domain specificity, annotation quality, and data diversity. This issue is pinpointed by Neves et al. [131] where they report an extended set of tools for biomedical annotation, a list of corpora with biomedical entities, and the types of annotations present in these corpora.

As shown in Huang et al. [66] annotations on different corpora may differ depending on the task tackled by each corpus. Across all corpora, the entity types include genes, proteins, mutations, diseases, disorders, drugs, chemicals, anatomy, cell types, pathways and process, signs and symptoms, diagnostic tests and measurements, procedures, and treatments. For QA models that take into account entities [198, 182] when computing an answer or QA models that depend on biomedical relational databases this can be an issue since there is no universal schema for biomedical entities and relations [130, 65, 175].

### Use of long sentences

In biomedical abstracts, long sentences can be found that include long complex sentences. The average character length of an abstract sentence in PUBMED is 151 characters[4] and the maximum number of characters is 2187 characters. The longest sentences contain mainly

[3]For example 'MELAS' is the abbreviated form of the longer sequence 'mitochondrial encephalomyopathy, lactic acidosis, and stroke-like episodes' which describes using one word the state of a patient.

[4]Statistics were computed in a sample of $1.06M$ sentences from $100K$ abstracts published on 2021 and contain both a title and an abstract.

extended lists with enumerated items or errors of the sentence tokenizer but even by removing them the average length in characters remains high (143 characters on average).

Examples of long sentences in biomedical abstracts are the following:

- 'Impaired social interaction, restrictive and narrow interests, anxiety, depression; aggressive, repetitive, rigid and self-injurious behavior, lack of consistency, short attention span, fear, shyness and phobias, hypersensitivity and rapid mood alterations, high level of food and toy selectivity; inability to establish friendships or follow the instructions; fascination by round spinning objects and eating non-food materials are common psychological characteristics of autism.'

- 'The procedures described here address a range of challenges in MarkVCID's design, notably: (1) acquiring all data under informed consent and enrollment procedures that allow unlimited sharing and open-ended analyses without compromising participant privacy rights; (2) acquiring the data in a sufficiently wide range of study participants to allow assessment of candidate biomarkers across the various patient groups who might ultimately be targeted in VCID clinical trials; (3) defining a common dataset of clinical and cognitive elements that contains all the key outcome markers and covariates for VCID studies and is realistically obtainable during a practical study visit; (4) instituting best fluid-handling practices for minimizing avoidable sources of variability; and (5) establishing rigorous procedures for testing the reliability of candidate fluid-based biomarkers across replicates, assay runs, sites, and time intervals (collectively defined as the biomarker's instrumental validity).'

**Multi-modality**

Multi-modality in biomedical question answering (QA) [229, 109, 80, 101, 123, 59]refers to the use of multiple modalities such as text, images, and tables in order to provide an answer to a question. The use of multiple modalities in biomedical QA makes it more difficult than generic domain multi-modal QA for several reasons. Images and tables often contain different types of information than text and may require different types of processing. For example, images may contain visual information about the location and size of a tumor, while tables may contain quantitative data about the patient's vital signs. Experiments have also been applied to radiology [83] and x-ray [122] data, using deep learning models to process biomedical images. Integrating this information from multiple modalities can be challenging and can lead to the increased complexity of the QA system.

### Limitations of Generative Models in Biomedical QA

Generative language models, such as the GPT-3 [25] and T5 [152] models, have received significant attention in recent years due to their impressive ability to generate high-quality, coherent, and grammatical text. These models can be trained on large datasets of text, allowing them to capture the underlying patterns and structures of natural language, and generate new text that is highly convincing and often indistinguishable from human-written text.

However, one growing concern with these models is the potential for them to produce hallucinations, or false and misleading information that is presented as if it were true. These hallucinations can occur even when the training data is entirely correct and unbiased. The root cause lies in the model's ability to combine expressions and information during text generation, which statistically align with each other and the given question but result in statements that lack coherence or validity.

This is a particularly concerning issue in the medical domain, where the use of inaccurate or misleading information could have serious consequences for patient health and safety. Generative language models trained on medical records or other health-related data could potentially generate false diagnoses or recommend inappropriate treatments, which could put patients at risk.

### Contradictory claims

Additionally, as the field of biomedical research continues to advance, it is not uncommon for scientific claims and findings to be later disproven or revised based on follow-up research. While a scientific article may present information that is believed to be true at the time of publication, further research may reveal that the initial findings were incorrect, incomplete, or based on flawed assumptions [7].

For example, if a biomedical question is asked that refers to a claim or hypothesis made in a scientific article from several years ago, it is important to consider whether there has been any follow-up research conducted that either supports or refutes that claim. Without this consideration, it is possible that the answer provided by a question-answering system could be inaccurate or out-of-date. Similarly, if a question is asked about a topic that is currently being actively researched, it is important to take into account the current state of knowledge and any ongoing studies that may provide new information. This can ensure that the answer provided by a question-answering system is as accurate and up-to-date as possible.

Therefore, when developing question-answering systems for the biomedical domain, it is important to take into account the temporal aspect of both the questions and the answers. This

can be achieved by incorporating up-to-date research findings and considering the potential for follow-up research that may refute or support previously accepted claims or hypotheses. Ultimately, this will lead to more accurate and reliable information being provided to medical professionals and patients alike.

In summary, biomedical NLP is a complex task due to the technical vocabulary, high level of abstraction, multi-modality, multilingualism, and privacy and ethical concerns. These characteristics make it different from the generic NLP domain and require specialized NLP models and techniques to accurately process and extract information from biomedical texts.

## 1.3 Outline of the thesis

In this thesis, we focus on improving existing deep-learning models and methods for information retrieval and question-answering in the biomedical domain.

Chapter two provides background information on NLP and machine learning methods that we use. We briefly describe some deep learning building blocks such as CNNs and Transformers that we later use throughout all chapters of the thesis. We also describe a strong traditional scoring function (BM25) that we use for document and snippet retrieval.

In chapter three we present details about all datasets available and already used to train QA and MRC models. We discuss any notable features of the datasets and how they have been used in previous research in the field. Overall, the aim of this chapter is to give a detailed and thorough understanding of the datasets available for training QA and MRC models, and how they have been used in the past.

In chapter four we present new deep-learning methods for biomedical document and snippet retrieval. We propose new deep-learning models that jointly retrieve snippets and documents and compare them with models that retrieve documents and snippets in a pipeline setting. The proposed models surpassed the state of the art and achieved first place in the BIOASQ challenge for the tasks of document and snippet retrieval in the sixth, seventh and eighth years of the competition and second place in the ninth year of the competition. Finally, we discuss the implementation of a retrieval system developed to aid biomedical experts in two real-life cases.

In chapter five biomedical machine reading comprehension (MRC) is discussed. We created two large biomedical MRC datasets using unsupervised methods previously used in generic domain data and make them available to the public. Strong deep-learning baselines have been proposed and a leaderboard has been created for the most recent of the two new datasets. Human evaluation conducted with the help of biomedical experts as well as non-experts shows the necessity of human expertise in order to answer biomedical questions.

It also shows that the proposed deep learning models yield great results and compete with human evaluators.

In chapter six we examine data augmentation (DA) techniques that can be used for biomedical factoid question answering. We compare seven data DA techniques that use pre-trained language models, machine translation, text generation, and MRC data including a dataset created in chapter four. All data augmentation techniques improve results in factoid QA using biomedical data. An extensive ablation study and further experiments show that the best method to augment data is to increase the context of the given text with surrounding sentences or alternatively using pre-trained language models to replace parts of the text.

In Appendix A extensive additional experiments and ablation tests are presented for retrieval models and data augmentation models. In Appendix B we present a preliminary but unpublished study for document retrieval using dense vector representations conducted during this thesis. We examine ways to create vector representations of snippets and questions and then use the vectors of the questions to retrieve relevant snippets using only proximity measures in the trained vector space. Preliminary results show that our approach surpassed at the time other deep learning models for both biomedical and general domain snippet retrieval.

# Chapter 2

# Background

The field of natural language processing (NLP) has seen a significant advancement in the development of deep learning models for various NLP tasks. In this chapter, we will focus on Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformer models, which have been proven to be highly effective for a wide range of NLP tasks and we use throughout this thesis. We will provide a detailed overview of each model, including their architecture, key features, and main variations. Additionally, we will present in detail the BM25 algorithm which is used in all of our retrieval models.

## 2.1 BM25

Given a pool of documents and a natural language query, a scoring function must be applied to rank all documents according to their relevance to the query. There is a plethora of scoring functions that can be used as scoring mechanisms in search engines such as Divergence From Randomness (DFR) [60], Divergence From Independence (DFI) [45], Information Based Model (IB) [100], LM Dirichlet [219], LM Jelinek Mercer [218], Relaxed Word Mover's Distance (RWMD) [86] and BM25 [157]. See Mitra and Craswell [120] for an overview of the methods. BM25 is widely adopted as a default scoring mechanism for many retrieval engines such as ElasticSearch[1], Solr[2] or Galago[3] and other search engines build on top of

[1] See `https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html` for more information on using BM25 with ElasticSearch.

[2] See `https://solr.apache.org/guide/7_0/learning-to-rank.html` for more information on using BM25 with Solr.

[3] See `https://lucene.apache.org/core/7_0_1/core/org/apache/lucene/search/similarities/BM25Similarity.html` for more information about using BM25 on Galago.

Apache Lucene[4]. In most retrieval models that we built, we focus on re-ranking a list of potentially relevant document obtained using BM25, therefore we explain the key parts of the BM25 function.

The BM25 score between a query $Q$ and a document $D$ is defined as:

$$\text{BM25}(Q,D) = \sum_{i}^{N} IDF(q_i) * \frac{tf(q_i,D) * (k_1+1)}{tf(q_i,D) + k_1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})} \tag{2.1}$$

where

$$IDF(q_i) = ln(1 + \frac{docCount - df(q_i) + 0.5}{df(q_i) + 0.5}) \tag{2.2}$$

BM25 takes as an input a query $Q$ (a set of tokens found in the user's question) and a candidate document $D$ (a set of tokens found in a text of the corpus) and computes a relevant score. In our experiments we use BM25 to score any document found in our corpus with respect to a user's question.

In the above equations, $tf(q_i,D)$ is the number of occurrences of the $i$-th query term $q_i$ in the document (term frequency), $df(q_i)$ is the number of documents in the collection which contain $q_i$ (document frequency) and *docCount* is the total number of documents. To reward matches that can be found in shorter documents (e.g. a title) the fraction $\frac{fieldLen}{avgFieldLen}$ is used, where *fieldLen* is the length of the document and *avgFieldLen* is the average length of all documents in the corpus.

$k_1$ is a parameter that helps to balance the importance of the term frequency in a document. It limits the amount of weight that a single query term can have on the score of the document. If $k_1$ is set to a high value, it allows a single term to have more weight on the score of a document, while if it is set to a low value, it reduces the weight of a single term on the score of the document. As the frequency of a term in a document increases, the score of the document will also increase, but at some point, the increase in score will start to saturate, meaning it will not increase as much for each additional occurrence of the term. The parameter $k_1$ is used to control this saturation point, by limiting the maximum effect that a single query term can have on the score of a document. By default, $k_1$ has a value of 1.2 in Elasticsearch[5] and Galago[6], the two search engines we have used throughout our research.

b is a parameter that controls the degree of length normalization in the BM25 algorithm. Length normalization is a technique used to adjust the score of a document based on its

[4]See `https://lucene.apache.org/core/7_0_1/core/org/apache/lucene/search/similarities/BM25Similarity.html` for more information on Apache Lucene and its implementation of BM25

[5]Visit `https://www.elastic.co/` for more details

[6]Visit `https://sourceforge.net/projects/lemur/` for more details

length. The idea behind this is that, in general, longer documents will contain more relevant information than shorter documents, but it also might contain more irrelevant information. Therefore, the length of a document should be taken into account when determining its relevance. $b$ usually ranges between 0 and 1. When $b$ is set to 0, the length normalization is not applied, and the scores are only based on term frequency and relevance. When $b$ is set to 1, the length normalization is applied to the fullest and the scores are heavily influenced by the length of the candidate document. By default, $b$ is equal to 0.75 for both Elasticsearch and Galago implementations.

## 2.2 Recurrent neural networks (RNN)

In this section, we describe the two types of RNN neural networks used during this thesis. We assume that the reader is familiar with linear layers and multi-layer perceptrons (MLP). Readers who are not familiar with neural network basics may wish to consider Russel et al. [160] or Goldberg et al. [54] first.

Recurrent Neural Networks (RNNs) are different from Multi-Layer Perceptrons (MLPs) in that (RNNs have a 'memory' component, allowing them to process sequential data. This memory component is implemented through the use of recurrent connections, which allow previous states of the network to be passed on to the current state. MLPs, on the other hand, are feedforward networks that do not have any memory component and process input data independently of previous inputs. This makes RNNs more suitable for tasks that involve processing sequential data such as time series, speech recognition, and natural language processing, while MLPs are better suited for tasks where the input data is independent of previous inputs such as image classification, and regression problems.

### 2.2.1 Long Short-Term Memory (LSTM)

An LSTM is one of the most used RNNs in deep learning. The core concepts of the LSTM are the cell state and its various gates. An LSTM can process a sequence input and it can store information in its cell state and update, forget or extract information using its corresponding 'gates' (Equations 2.3).

In an LSTM, the update, forget, and input gates control the flow of information through the network. The input gate ($i_t$ in Equation 2.3) controls how much new information is added to the current memory state. The update gate ($C'_t$ in Equation 2.3) controls how much of the previous memory state is passed on to the current state. The forget gate ($f_t$ in Equation 2.3) controls how much of the previous memory state is forgotten. The output gate ($O_t$ in

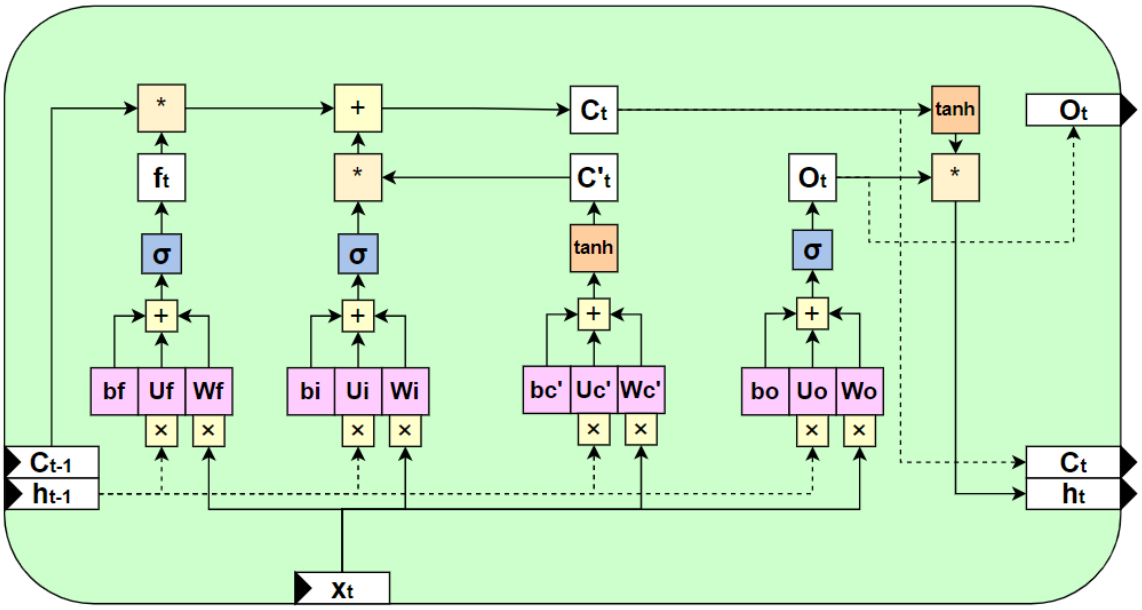


Fig. 2.1 The LSTM architecture. The outputs of the previous steps $C_{t-1}$ and $h_{t-1}$ are used as input to the LSTM. Equations 2.3 and 2.4 are used to compute the output of the current timestep $O_t$ to be used for the downstream task and the $C_t$ and $h_t$ results are forward propagated to be used in timestep $t+1$. Since $C_t$ and $h_t$ are used at the next timestep the trainable parameters $W_t$, $W_f$, $W_{C'}$, $W_o$, $U_t$, $U_f$, $U_{C'}$, $U_o$, $b_t$, $b_f$ and $b_{C'}$ will be updated with back-propagated loss from all $O_t$, $C_t$ and $h_t$.

Equation 2.3)controls the output of the LSTM cell. It takes in the current input, the previous hidden state, and the current cell state and produces a new output, which is then used to update the hidden state. Together, these gates allow the LSTM to selectively choose which information to keep, which information to discard, and which new information to incorporate, enabling the network to maintain a long-term memory of the input sequence.

$$\begin{aligned} f_t &= \sigma(W_f \times x_t + U_f \times h_{t-1} + b_f) \\ i_t &= \sigma(W_i \times x_t + U_i \times h_{t-1} + b_i) \\ O_t &= \sigma(W_o \times x_t + U_o \times h_{t-1} + b_o) \\ C'_t &= tanh(W_{C'} \times x_t + U_{C'} \times h_{t-1} + b_{C'}) \end{aligned} \tag{2.3}$$

Equations of the LSTM can be seen in Equation 2.3. The LSTM computes a candidate memory ($C'_t$) and combines it (Equation 2.4) with the memory produced in the previous step ($C_{t-1}$) to form the current state memory ($C_t$). The output of the LSTM at timestep $t$ is denoted as $O_t$ (Equation 2.4). $C_t$ along with $h_t$ carry information about the already processed part of the sequence and are provided as input to the next timestep.

$$
\begin{aligned}
C_t &= f_t \odot C_{t-1} + i_t \odot C'_t \\
h_t &= tanh(C_t) \odot O_t
\end{aligned}
\tag{2.4}
$$

$W_t$, $W_f$, $W_{C'}$, $W_o$, $U_t$, $U_f$, $U_{C'}$, $U_o$ are trainable matrices and $b_t$, $b_f$ and $b_{C'}$ are trainable scalars of the LSTM. At time-step 1, $C_0$, and $h_0$ are randomly initialized or initialized as zero vectors.

### 2.2.2 Gated Recurrent Unit (GRU)

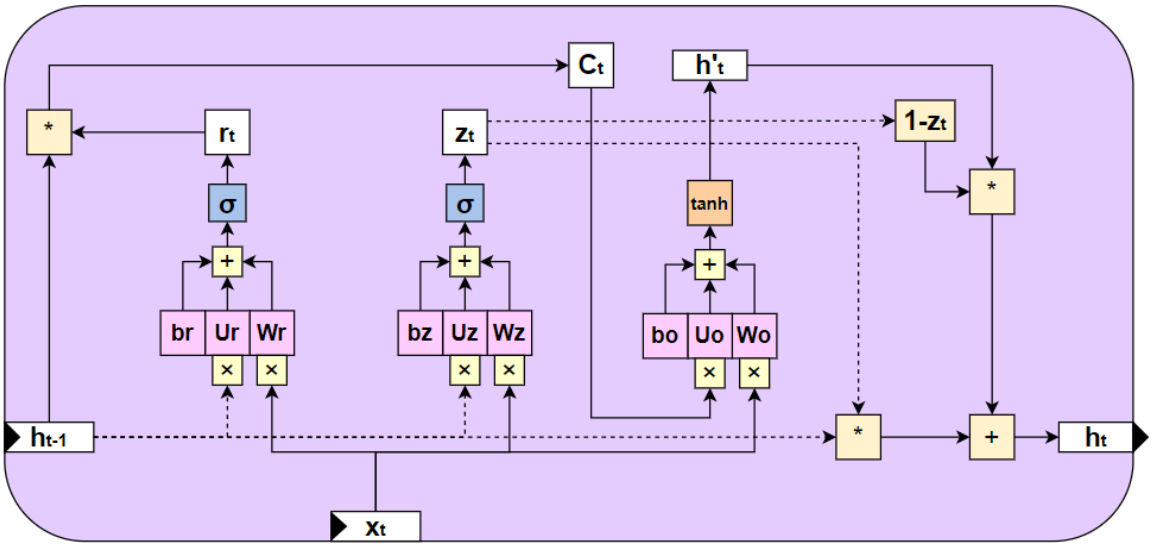


Fig. 2.2 The GRU architecture. The output of the previous step $h_{t-1}$ is used as input to the GRU unit. The Equations 2.5 and 2.6 are used to compute the output of the current timestep which is forward propagated to be used in timestep $t+1$. Since $h_t$ is used at the next timestep the trainable parameters $W_r$, $W_z$, $W_o$, $U_r$, $U_z$, $U_o$, $b_r$, $b_z$ and $b_o$ will be updated with back-propagated loss from $h_t$.

A simpler RNN structure is the Gated Recurrent Unit (GRU). As seen in Figure 2.2 and Equation 2.5, the output of a GRU is simply a hidden state at the current step $h_t$. Two gates ($r_t$ and $z_t$ in Equation 2.5) decide how much of the past information to forget and how much information of the input should be kept. The scalars computed by the two gates are used to weight and sum the output of the previous steps ($h_{t-1}$) and the current step's output $h'_t$ to create the final output of the current step $h_t$ (Equation 2.6).

$$
\begin{aligned}
r_t &= \sigma(W_r \times x_t + U_r \times h_{t-1} + b_r) \\
z_t &= \sigma(W_z \times x_t + U_z \times h_{t-1} + b_z) \\
h'_t &= \tanh(W_o \times x_t + U_o \times C_t + b_o)
\end{aligned} \tag{2.5}
$$

$$
h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{2.6}
$$

$W_r$, $W_z$, $W_o$, $U_r$, $U_z$, and $U_o$ are trainable matrices and $b_r$, $b_z$ and $b_o$ are trainable scalars of the GRU. They are randomly initialized and are updated through back-propagation during training. At time-step 1, when the first input of the sequence is processed, $h_0$ is randomly initialized or initialized as a zero vector. GRU has fewer trainable parameters than LSTM which makes it quicker to train and demands less computational resources when processing the same sequence.

## 2.3 Convolutional neural networks (CNNs)

Convolutional neural networks (CNN) are a type of artificial neural network that uses a mathematical operation called convolution instead of general matrix multiplication used in MLPs. In a convolution, filters (weight matrices) are used to multiply a part of the input feature map and compute an output. Convolution were originally used to process images using manually crafted filters.

Convolutional layers allow for parameter sharing, which means that the same weights are used for multiple locations in the input. Therefore they have the ability to recognize patterns regardless of their position in the input. They also reduce the number of trainable parameters, make the model more efficient, and allow the parallelization of the convolutions. Convolutional layers also use local connectivity, meaning that each neuron is only connected to a small region of the input. This allows for more efficient learning and better generalization.

Convolutional layers in natural language processing can help to capture the local dependencies between words in a sentence. This is done by applying a convolutional filter over the input sentence, which is then used to extract local features from the text. This helps to identify local patterns and features in the text and can be used, for example, to detect sentiment, classify topics, and identify entities. Additionally, convolutional layers can help reduce the number of trainable parameters in a model, as they are able to capture more complex relationships between words using fewer parameters than recurrent neural networks. This makes convolutional layers more efficient and better suited for large datasets.

A 3-dimensional convolutional neural network accepts an input matrix $X$ of size $H_{input} \times W_{input} \times D_{input}$ where $W$, $H$, and $D$ stand for width, height, and depth respectively. Tradition-

ally, $K$ randomly initialized filters of size $F_H \times F_W \times D_{input}$ are created. The filters are then placed on top of the input matrix and are gradually moved from top to bottom and left to right. For each filter $M$ an output matrix $O$ is computed using Equation 2.7 to compute the element of the output matrix at row $i$ and column $j$, where $b$ is a trainable randomly initialised scalar.

$$O_{i,j} = \left( \sum_{a=0}^{F_H-1} \sum_{b=0}^{F_W-1} \sum_{c=0}^{D_{input}-1} X_{a+i,b+j,c} * M_{a,b,c} \right) + b \tag{2.7}$$

Each time the filters move by a step size which is traditionally called stride. In Equation 2.8 the step used to move the filter along the height axis is denoted as $S_H$ while $S_W$ corresponds to the step applied along the width axis. By reducing the stride, the output feature maps will have a higher resolution, whereas increasing the stride will result in lower resolution output feature maps. Striding is mainly used to reduce the computational cost of the network.

Padding is used in CNNs to maintain the spatial dimensions of the input while applying convolutional filters. Without padding, the spatial dimensions of the input will be reduced with each convolutional layer, which can lead to a loss of spatial information and decrease in performance. Additionally, padding can also help to reduce the amount of border effect that can occur when applying convolutional filters to the edges of an image. Therefore the input of the convolution is expanded by adding vectors with zero values at the borders of the input. $P_H$ vectors are added at the top and the bottom of the input and $P_W$ vectors are added on the left and the right borders of the input.

In each step, a dot product is computed between the values of the filters and the part of the input matrix that overlaps with the filter. [7] The output of the convolution is a matrix of size $W_{output} \times H_{output} \times D_{output}$ where:

$$\begin{aligned} H_{output} &= (H_{input} - F_H + 2P_H)/(S_H + 1) \\ W_{output} &= (W_{input} - F_W + 2P_W)/(S_W + 1) \\ D_{output} &= K \end{aligned} \tag{2.8}$$

Filter dilation in CNNs is a technique used to increase the receptive field of a convolutional filter without increasing the number of parameters. It is achieved by inserting zeroes between the elements of the filter kernel and then convolving it with the input image. This increases the spacing between the elements of the filter, effectively increasing the area of the input that the filter can "see" or be receptive to. This technique can be useful in tasks such as image segmentation, where a larger receptive field is needed to capture fine-grained details in the image.

[7] Both striding and dot product operations can be parallelized thus making the CNN architecture faster than RNN's.

In convolutions applied to textual data, the filters are usually applied to the embedding of the tokens. In that case, as presented in Figure 2.3, the second dimension of the filters is usually equal to the size of the tokens' embeddings ($F_H \times F_W \times D_{input}$ is equal to $F_H \times E \times 1$ where $E$ is the size of the embedding of the token).

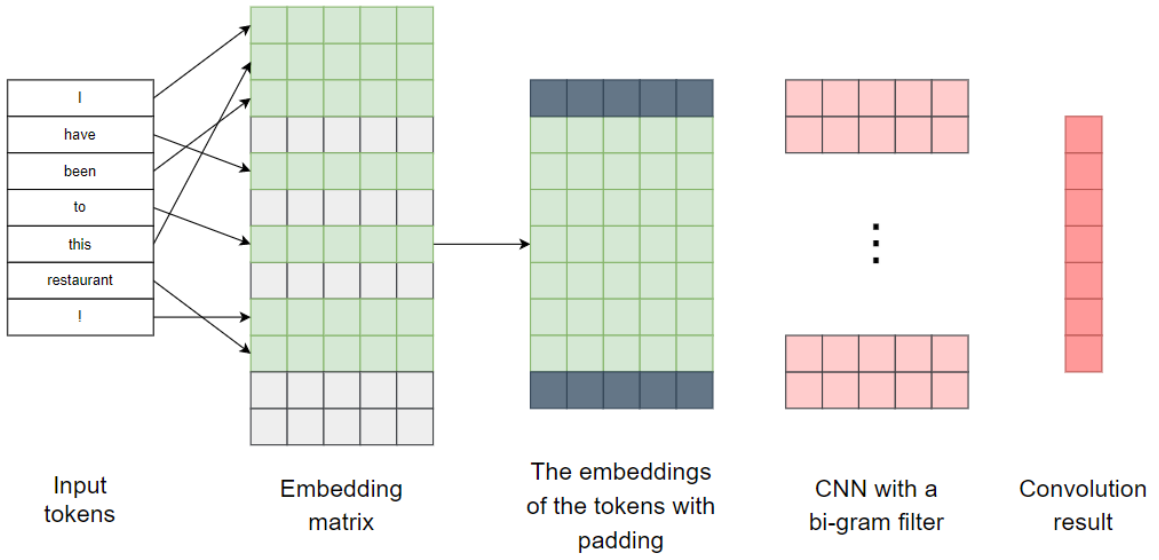


Fig. 2.3 Process of a CNN using one bigram filter. Word embeddings are retrieved for all input tokens (green color). Light grey boxes in the embedding matrix (left) depict word embeddings not detected in the input text. Padding is applied (dark grey boxes in the middle) to the embeddings of the tokens. Starting from top to bottom a slice of two rows is selected from the green matrix and a dot product is computed between the slice and the randomly initialized trainable filter (pink boxes) to compute one cell of the convolution output (a red box). Then the next two rows are used to compute the next red box and this process is repeated until all rows are used.

In our experiments, we used CNNs specifically tailored for text processing. As presented in Figure 2.3 given a sequence of tokens a feature map is computed by selecting the embedding of each token. To apply a convolution we use a filter of size $K \times E$ where $K$ is the number of tokens we want to examine simultaneously and $E$ is the size of the word embeddings. We drag the filter on top of the feature map and apply an element-wise multiplication and sum the output scores to extract a score for the examined words. In Figure 2.3 a bigram convolution is applied ($K = 2$) so the filter is applied to the vectors of two sequential tokens each time. A higher $K$ would mean that we use more words in each step but the trainable parameters of the model grow.

This process can be applied multiple times with a different filter each time as in Figure 2.4. For each filter, a separate score is computed and the output of all filters combined creates

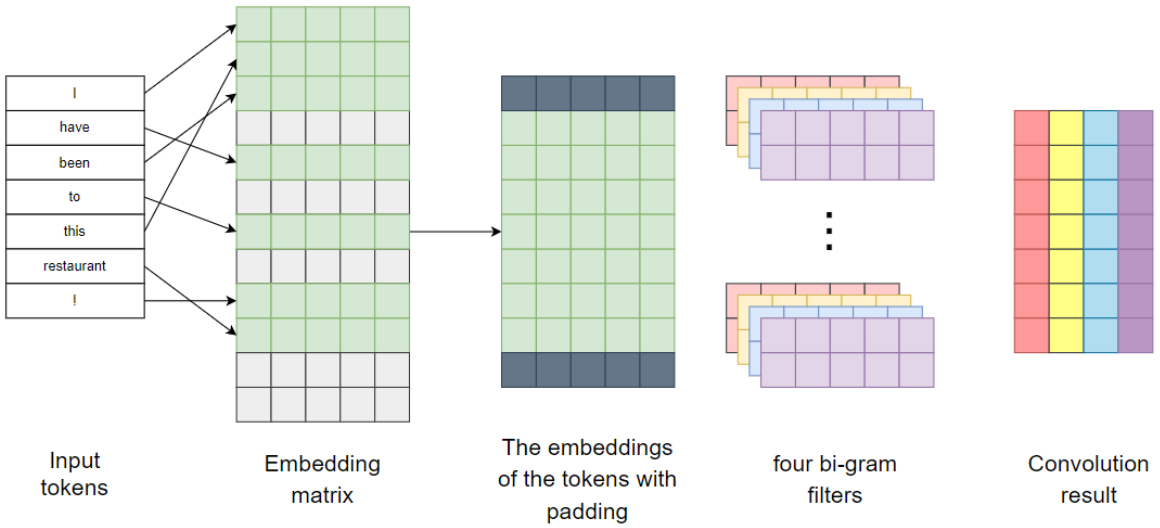


Fig. 2.4 Process of a CNN using four bigram filters. The process resembles Figure 2.3. Each filter is depicted with different color and the results of the convolutions are concatenated to form the final result. Ideally, each filter captures a different feature of the consumed text.

a contextual embedding of the word as it has used features from the context (surrounding words). The number of applied filters as well as the size of the filters are hyper-parameters set by humans before training.

After a convolution, a contextual embedding is computed for each token. In some experiments where a embedding for the entire sequence is needed, pooling methods are applied. In Max-pooling the maximum score of each column is selected as an element for the embedding of the entire sequence (see Figure 2.5). Similarly Average-Pooling the average of all elements in the column is computed. More complex functions have also been proposed as pooling mechanisms but will be further explained in our proposed models.

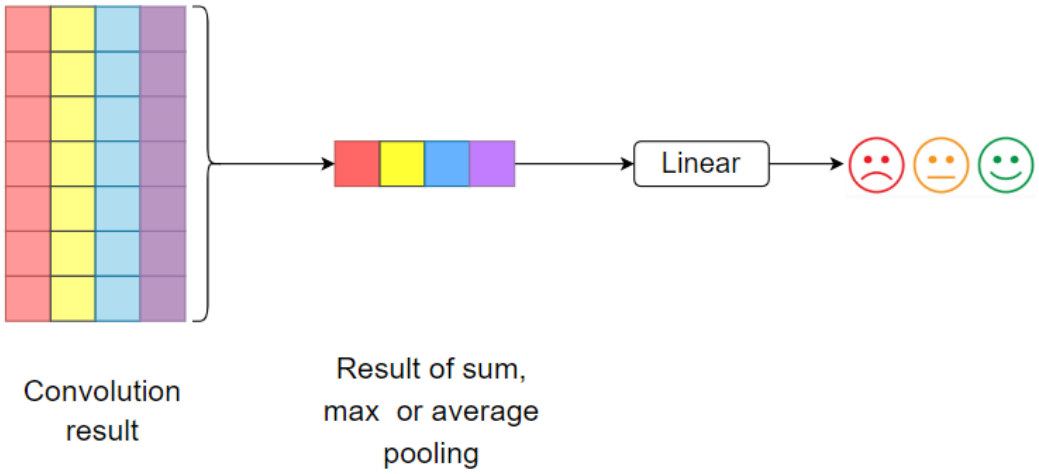


Fig. 2.5 Pooling on the result of the convolution. In this example, the resulting feature vector of the pooling mechanism is used as an input to a linear classifier for sentiment classification.

## 2.4 Transformer-based neural networks

The transformer architecture [186] sparked the creation of most state-of-the-art models in Natural Language Processing. In this section, we present a quick overview of the transformer architecture.

### 2.4.1 Byte-Pair Encoding (BPE)

Most transformer-based neural networks operate on sub-word units. One of the most common algorithms for the extraction of sub-word units is the Byte-Pair encoding of tokens. Byte-Pair Encoding [165] is a subword-based tokenization algorithm that splits the rare words into smaller meaningful subwords (e.g. 'subword' turns to two subword units 'sub' and '##word' where '##' indicates that the token 'word' completes the preceding token). Each subword unit is called a BPE. Given a pre-defined size of $k$ BPEs for the vocabulary the algorithm identifies the $k$ most common sub-sequences of characters in a big corpus and adds them to the vocabulary.

As shown in Table 2.1 given an example sequence 'teaching deep learning' as our corpus the algorithm identifies the most common pair of characters 'in' and replaces it with a new unseen character 'I'. The algorithm repeats three times replacing 'ea' with 'E' and 'Ig' with 'G' and stops when the replacement matrix has $k$ items (i.e. the vocabulary size has reached the maximum allowed BPEs).

Using BPE, we start by finding the most common character pair in the sequence and merge them into a single token. This process is repeated until the desired vocabulary

| **Sequence alterations** | **Steps** |
|---|---|
| teaching deep learning | replace ‘in’ with **I** |
| teach**I**g deep learn**I**g | replace ‘ea’ with **E** |
| t**E**chIg deep l**E**rnIg | replace ‘Ig’ with **G** |
| tEch**G** deep lErn**G** | final representation |

| Byte Pair | Replacement |
|---|---|
| Ig | G |
| ea | E |
| in | I |

Table 2.1 Example of BPE construction of a random sequence of characters. On the left part of the figure, we observe how the sequence changes gradually as we replace ‘in’ with ‘I’, then ‘ea’ with ‘E’, and ‘Ig’ with ‘G’. The replacements are stored in the vocabulary on the right. When millions of texts are processed the vocabulary includes the most common sequences of characters along with the original unique characters of the processed texts.

size or a maximum number of iterations is reached. This way, the model can learn better representations of the words that appear in the text, even if they are rare words or out-of-vocabulary words.

### 2.4.2 Self Attention

The most important feature of the transformer block is the self-attention mechanism. In a transformer block, a Self-Attention module computes an importance score for each sub-word unit indicating how much the embedding of one sub-word unit contributes to the calculation of the contextual embedding of every other sub-word unit in the input sequence.

In order to compute the attention scores the embeddings of the input sub-words, denoted in Figure 2.6 as $X$, are multiplied by two trainable weight matrices $W_Q$ and $W_K$. The notation $Q$ stands for Query and those vectors act as ‘givers’ i.e. they encode the information. The notation $K$ stands for Key and those vectors act as ‘receivers’ i.e. they receive information from the $Q$ vectors.

As seen in Figure 2.7 and Equation 2.9 the $Q$ matrix is multiplied by the transpose of the $K$ matrix to form a cross-attention matrix. The result is then normalized using the dimensionality of the vectors $\sqrt{d_k}$ and softmax is applied on each row of the attention matrix so that the values of each row sum to 1.0. In the final attention matrix, the cell in row $i$ and column $j$ holds an attention weight that indicates how much should the vector of the sub-word at position $j$ be taken into account when computing the contextual representation of the sub-word at position $i$.

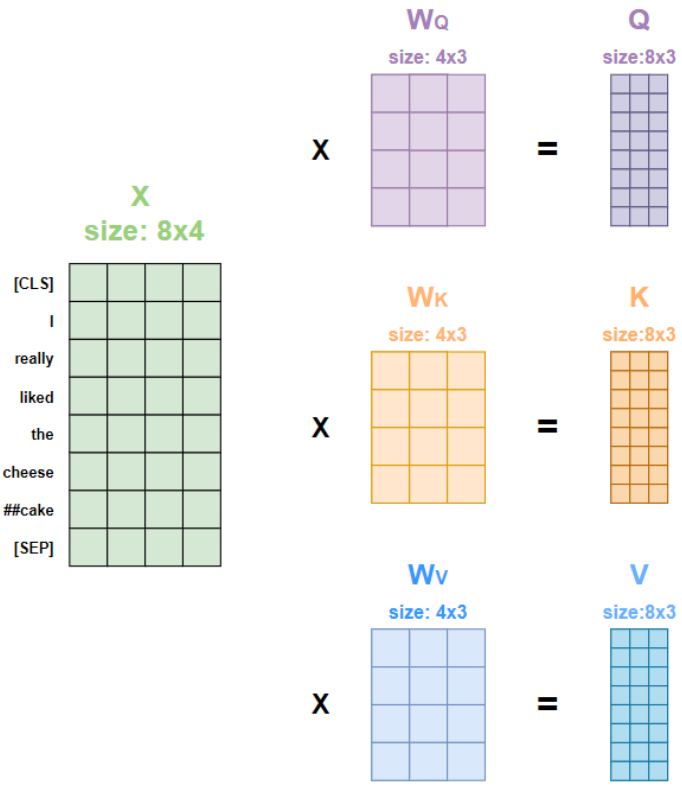


Fig. 2.6 Self-Attention computation of Key-Query and Value vectors for 8 sub-words. The embeddings of the sub-words are multiplied with three matrices containing trainable weights. The $W_v$ matrix is used to compute the output of a traditional Dense layer as if no attention would be applied on top of the results. $W_Q$ and $W_K$ are trainable matrices used to compute $Q$ and $K$ matrices which are essential to the computation of the attention scores of Figure 2.7.

$$
\begin{aligned}
Q &= X \times W_q \\
K &= X \times W_k \\
V &= X \times W_v \\
Attention &= softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \\
A &= Attention \times V
\end{aligned}
\tag{2.9}
$$

In Figure 2.7, the darker color in the attention matrix indicates a higher value. The result of the multiplication $X \times W_v$ corresponds to the output of a traditional perceptron layer that we would use in a traditional MLP where no attention mechanism is applied. The attention matrix is multiplied to the $V$ matrix to extract contextual representations for each sub-word unit and the resulting matrix $A$ contains the contextual representations of the sub-word units. These

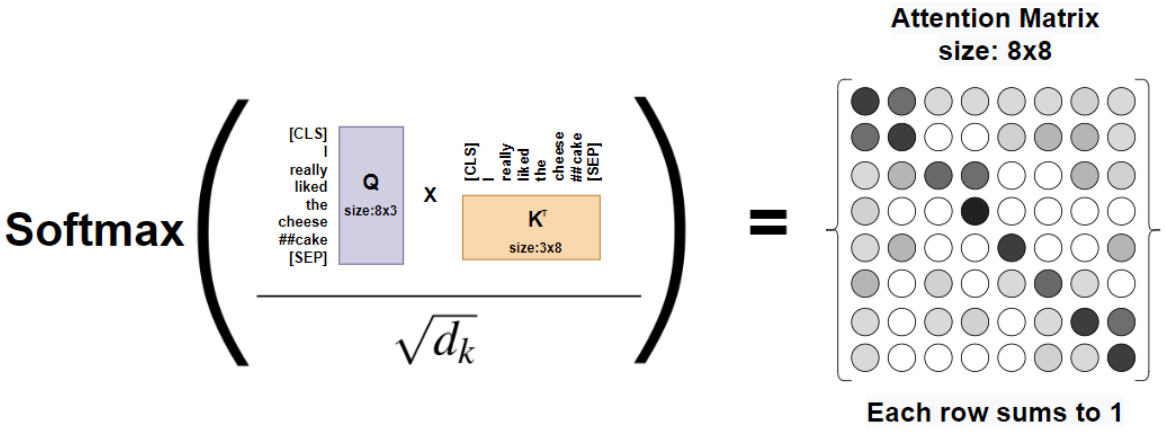


Fig. 2.7 Self-Attention computation of attention matrix for 8 sub-words. Darker color in the attention matrix indicates higher attention scores. The attention matrix is then used as in Figure 2.8 to apply attention on the results.

representations encode each sub-word but alter the representation using the representation of all sub-words in the input sequence.

Just like multiple filters can be used in convolutional layers aiming to capture different higher level features of the input, in a transformer block multiple attention layers can also be used to extract different contextual representations for the input sequence. In this process called Multi-Head Attention the $K, Q, V$ matrices of each attention mechanism are initialized with different weights for each separate self-attention instance (also called attention head). In Figure 2.9 we observe the output of 4 attention heads. In a transformer block the outputs of all attention heads are concatenated and the resulting matrix is then multiplied with a trainable weight matrix $W_{out}$. The $W_{out}$ is trained to combine all contextual embeddings of the sub-words and form the final contextual representations of the sub-words in matrix $Z$.

Many attention mechanisms have been proposed as alternative attention mechanisms for a transformer layer. For example ANNA model [75] introduced a Neighbor-aware Self-Attention mechanism which forces the diagonal of the attention matrix to have zero values. This way they enforce the computation of contextual representation of a sub-word unit using only the rest sub-word units of the input (i.e. the output of the transformer for a given sub-word depends only on the context and not the sub-word unit itself). ANNA at the time it was published achieved state-of-the-art results for Question Answering in SQUAD V1.1 dataset.

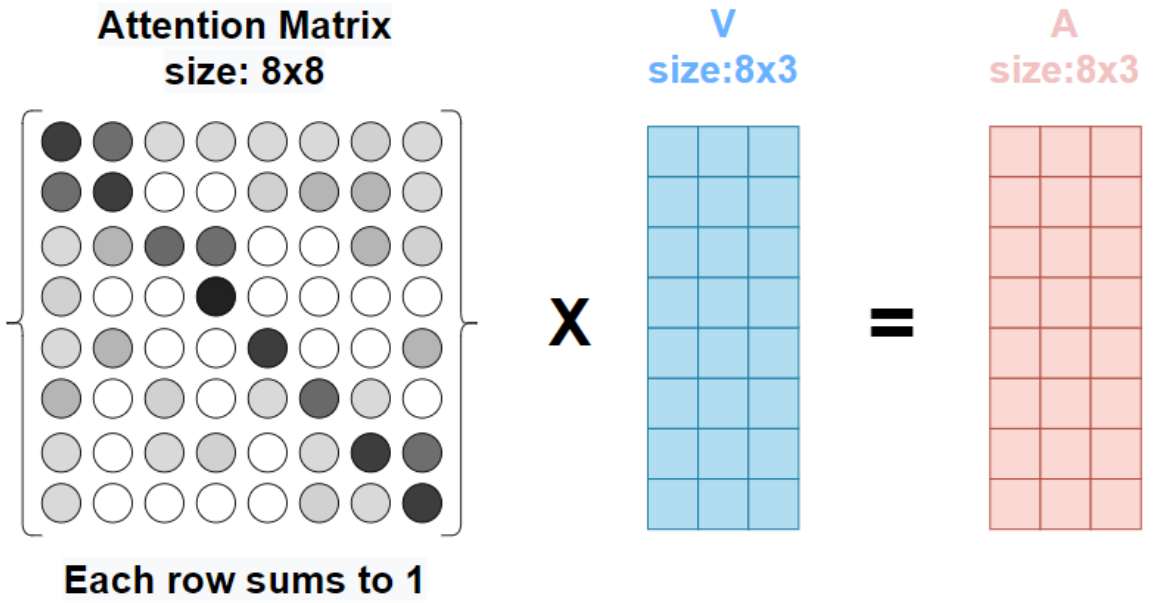


Fig. 2.8 Self-Attention computation on a transformer layer for 8 sub-words. The attention matrix is multiplied by the $V$ matrix. The matrix $V$ corresponds to the output of a traditional Dense layer.

### 2.4.3 Positional Embeddings

As one could see in the example of Figure 2.6, if we swap the sub-word 'I' in the second position with the sub-word 'liked' in the fourth position the only thing that would change in the output representations would be the position of the contextual representations. The contextual representations in matrix $Z$ in Figure 2.9 would also swap but the numerical values of the contextual representations would remain the same. Additionally, if the transformer consumed a long text, the contextual representations for identical sub-words would be the same regardless of their position in the text and their neighbouring tokens.

For these reasons, trainable positional embeddings are constructed and added to the input vectors of the transformer. The sub-word at position 1 is added to the embedding of position 1 and so on. These positional embeddings are fine-tuned during training along with the rest of the trainable parameters of the transformers. However, when positional embeddings are used a maximum sequence length (in sub-words) has to be set. In most cases in the bibliography, so far 512 positional embeddings are used setting a limit to the transformer input of 512 sub-words so longer sequences have to be truncated.

The original function used to compute positional embeddings in a transformer layer is the sinusoidal function [186] presented in Equation 2.10. The sinusoidal positional encoding outputs a unique encoding for each time-step (word's position in a sentence) in a

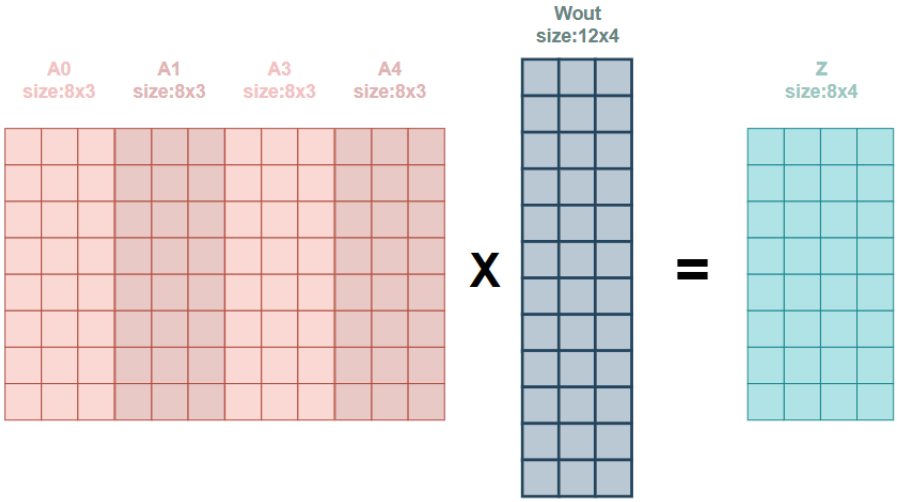


Fig. 2.9 The output of a Multi-Head Attention for 4 self-attention heads (pink color) is multiplied by a trainable output matrix $W_{out}$ (dark blue color) and the final output of the transformer layer is computed.

sequence. Since no trainable embeddings are constructed this function can produce positional embeddings for sentences regardless of their length.

$$PE_{(pos,i)} = \begin{cases} sin(\frac{pos}{10000^{2i/d}}), & if \quad i \mod 2 == 0 \\ \\ cos(\frac{pos}{10000^{2(i-1)/d}}), & if \quad i \mod 2 == 1 \end{cases} \tag{2.10}$$

Since unlimited positional embeddings can be computed using the sinusoidal function, a transformer layer is limited only by computational resources. As longer sequences are used as input to the transformer layer, bigger attention matrices must be used to compute the contextual representations. Several deep learning models have been proposed that process longer input sequences using fewer computational resources such as LongFormer [17], BigBird [215] or Smith [203].

The Longformer [17] is a transformer-based model that is designed to process long sequences in an efficient way. The proposed attention mechanism of Longformer scales linearly with the sequence length, making it possible to process documents of thousands of tokens or longer without the need to chunk or shorten the input.

Similar to LongFormer, BigBird [215] is a sparse attention mechanism that is linear in the number of tokens. The authros claim that the model achieves state-of-the-art performance on a number of natural language processing (NLP) tasks, such as question answering and long document classification. The authors also introduce a novel application of attention-

based models for extracting contextual representations of genomic sequences, like DNA, and fine-tuning it for downstream tasks in genomics.

SMITH [203] is a novel Siamese Multi-depth Transformer-based Hierarchical Encoder for document matching, which contains several design choices to adapt self-attention models for long text inputs. The model is pre-trained using the masked sentence block language modeling task to capture sentence level semantic relations within a document. Given a long document, SMITH first encodes the sentences of the document using an encoder block and then uses a sentence encoder block which consumes the embeddings of the sentences to encode the entire document. Eve though SMITH bypasses the length limitations of the input, it can only be used for document classification. The experimental results on several benchmark data show that the SMITH model outperforms the previous state-of-the-art models and increases the maximum input text length from 512 to 2048 when comparing with BERT based baselines.

### 2.4.4 The encoder transformer block

The architecture of the encoder transformer block can be seen on the left part of Figure 2.10.[8] The input of a traditional transformer block is sub-word units (Section 2.4.1) which are translated into sub-word unit embeddings using an embedding matrix. Positional embeddings (Section 2.4.3) are added to the sub-word unit embeddings to make them aware of their position. Self attention (Section 2.4.2) is applied to compute contextual representations for the sub-word units. A residual mechanism (i.e. the input of the self-attention layer is added to the output) is used to allow back propagation of loss even if all weights of the self-attention layer turn to zero and layer normalization normalizes the output values. A Feed Forward layer is then applied to each contextual vector representation. This layer is called time-distributed as the same layer is applied to all time-steps (all sub-words) of the input. Finally a second residual mechanism and layer normalisation is applied to get the output of the encoder transformer block.

### 2.4.5 Encoder - Decoder transformer

A transformer block as described above constitutes an encoder block. However, there are architectures and tasks, such as translation, language modeling, or text summarization, that also require decoding of an embedding or generating new text sequences. To that end, a decoder transformer block was also introduced[186].

[8] Figure 2.10 is retrieved and used under CC-by-4 license from 'Alammar, J (2018). The Illustrated Transformer.' : `https://jalammar.github.io/illustrated-transformer/`

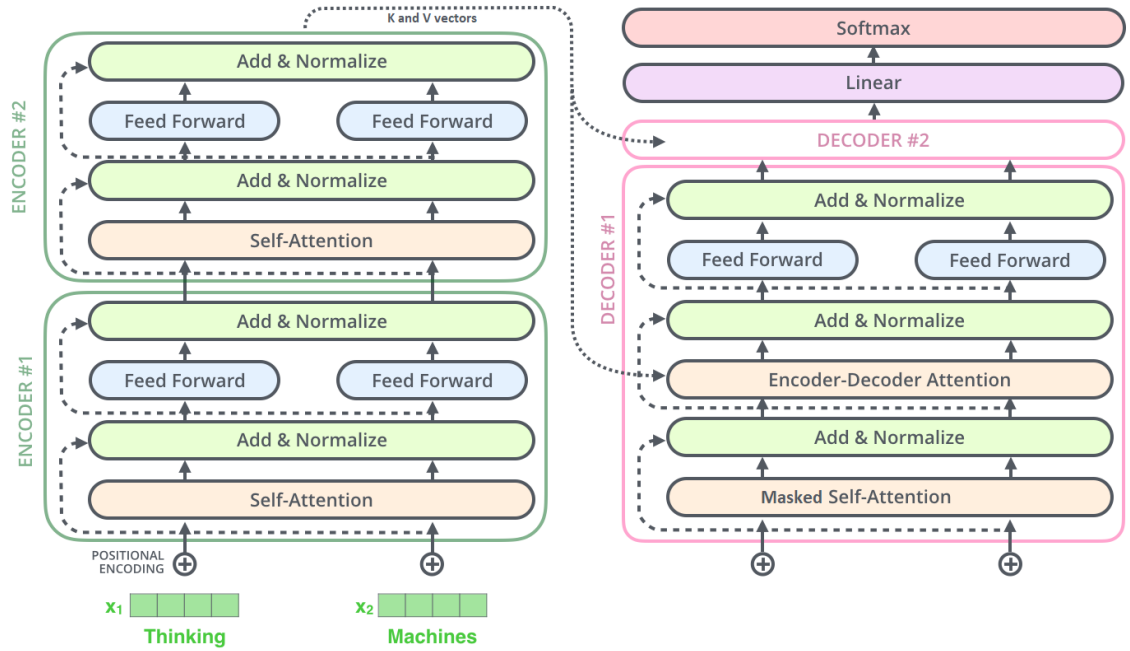


Fig. 2.10 An encoder-decoder architecture that uses both encoder and decoder transformer blocks. On the left of the figure (green boxes), two encoder blocks are presented, and on the right part (red box) the structure of a transformer decoder block is presented. The encoder block consumes embeddings of sub-word units and applies positional encoding. A Self-Attention mechanism computes contextual embeddings for the input sub-words. Then a residual mechanism is applied followed by layer normalization. Finally, a time-distributed Feed Forward layer is applied and a final residual and layer normalization layer computes the output of the transformer. The decoder block resembles the encoder block however the self-attention mechanism masks the input of the next timesteps when computing the output at timestep $t$. Additionally, a second attention layer is added which instead of computing its own $K$ and $V$ vectors uses the vectors computed by the encoder block.

The decoder transformer block operates differently than the encoder. The Masked self-attention layer in a decoder (See also Figure 2.10) is only allowed to attend to earlier positions of the output sequence by masking future positions before the softmax step in the self-attention calculation is applied. Then residual mechanism and layer normalization is applied. Additionally as a second differentiation from the encoder layer the 'Encoder-Decoder Attention' layer is used. The'Encoder-Decoder Attention' is similar to multi-headed self-attention, but instead of computing new key and value matrices it uses the key and value matrices (matrices $K$ and $V$) of the encoder layer. Therefore the output of the attention is a representation of both the input of the decoder the input sequence of the encoder block.

The last three steps of the decoder transformer block are similar to an encoder block. A residual mechanism and layer normalization are applied, followed by a time-distributed

linear layer. Finally a third residual mechanism and layer normalization is used to compute the output of the decoder transformer block.

### 2.4.6 Pre-Training tasks

Most recent state-of-the-art models for NLP use transformer-based architectures. These architectures are named and distinguished based on the number of layers as well as the pre-training tasks used. For example, BERT [41] is a transformer-based model, where the pre-training tasks are Masked Language Modeling (MLM), where the model is required to restore a token replaced by a special token '[MASK]' in the original input, and Next Sentence Prediction (NSP) where the model should decide whether a sentence follows another sentence in the original corpus they were sampled from or it was picked randomly. In BART, [98] the pre-training tasks would be Masked Language Modeling, Token Deletion, Text Infilling, Document Rotation, and Sentence Permutation using transformers in a Denoising Autoencoder architecture.

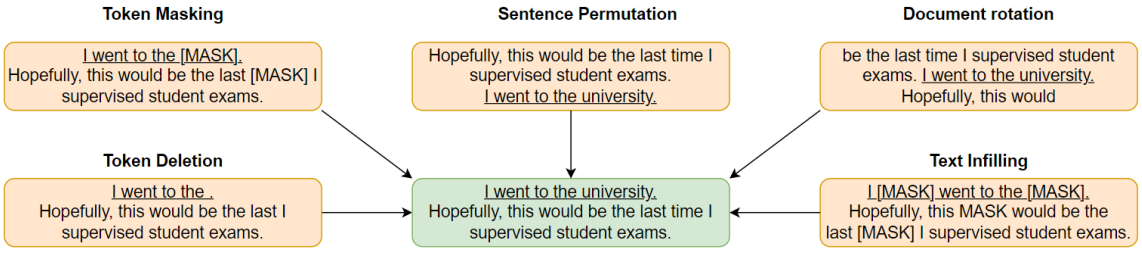


Fig. 2.11 An example of the tasks used to pre-train the original BART model.

An example of all pretraining tasks can be seen in Figure 2.11. The target output is presented in green color and includes two sentences (The tokens of the first sentence are always underlined to easily detect them). In 'sentence permutation' the sentences are shuffled and the model is trained to produce the original text. Similarly in 'document rotation' the original text is split into a random index and the order of the two parts of the text is changed. 'Document rotation' is identical to 'sentence permutation' when the random index coincides with the end of the first sentence. BERT-based models use token masking for masked language modeling where subword units are replaced with '[MASK]' tokens and the model is pre-trained to replace the '[MASK]' tokens with the missing subword units. In an encoder-decoder setting like BART, the decoder can produce more than one subword units to replace the '[MASK]' token therefore multiple subwords may be replaced with one '[MASK]' token in the input of the encoder. When training BART, besides using 'token masking', the authors also used 'token deletion' where the subword units are removed but

not replaced with the special token '[MASK]', and also use 'text infilling' where '[MASK]' tokens are randomly placed in the text. The model is pre-trained to ignore unnecessary '[MASK]' tokens, and also add subword units even if the '[MASK]' tokens are missing.

# Chapter 3

# Background on QA Datasets

## 3.1 Question Answering Datasets

In this thesis, we mainly focus on standalone questions and textual data associated with scientific biomedical literature. Our main goal is to develop advanced information retrieval and question-answering systems that can effectively extract relevant information from large biomedical corpora. In particular, we focus on the tasks of information retrieval and question-answering in the biomedical domain.

These tasks are crucial for helping biomedical researchers access the information they need to advance their work and improve human health. Therefore we provide information about datasets that can be used for the tasks of information retrieval, snippet extraction, and question answering.

## 3.2 Biomedical domain datasets

The most suitable and open dataset for Question Answering in the Biomedical Domain is the dataset of the BIOASQ challenge [184]. The BIOASQ challenge is an annual competition that evaluates the performance of information retrieval and question-answering systems for biomedical research. BIOASQ provides, among other information, human-annotated tuples of questions, relevant documents, and relevant snippets inside the relevant documents. These data are created by biomedical experts and can be used in order to train and evaluate QA systems.

In Task B phase B of the BIOASQ challenge, systems are given a set of questions and a set of relevant documents that have been retrieved in response to those questions. The

systems are then required to answer the questions by extracting the answer from the relevant documents. There are four different types of questions that require different types of answers.

A *factoid question* is a question that can be answered with a short, factual statement. These questions typically ask for specific information about a particular topic or entity. For example, "Which protein has been found to interact with phospholamban (PLN) and is also an anti-apoptotic protein?" are factoid questions. The answer to these questions is a single, verifiable fact or piece of information. An example of a factoid question can be found in table 3.1.

A *list question*, on the other hand, is a question that requires a list of items or entities as an answer. These questions typically ask for multiple pieces of information or a set of entities that are related to a particular topic. For example, "Which factors activate zygotic gene expression during the maternal-to-zygotic transition in zebrafish?", "What are the effects of depleting protein km23-1 (DYNLRB1) in a cell?" are list questions. The answer to these questions is a list or a set of items.

A *"yes/no" question* is a type of question that can be answered with a simple "yes" or "no" response. These questions typically ask whether a statement or claim is true or false. For example, "Is Weaver syndrome similar to Sotos?", "Are ultraconserved elements often transcribed?" are two examples of "yes/no" questions. Yes/No questions are used to evaluate the ability of information retrieval systems and QA systems to determine the veracity of statements or claims.

A *summary question* is a type of question that requires a summary or a brief overview of a set of documents. These questions are typically answered with a short text that provides a general overview of the main points or key takeaways from the relevant snippets provided by human experts.

The models submitted by contestants in BIOASQ are evaluated in batches (sets) of questions. The answers generated by the Participants' algorithms are evaluated based on various metrics such as precision, recall, F1 score, and MAP, and the results are used to rank the participants and compare the performance of different algorithms. Additionally, human experts review the submitted results to identify correct answers that may be missing from the gold corpus thus the final comparison of the submitted models is fair. Up to this date, 5 batches of evaluation data are published each year.

Another useful dataset in the biomedical domain is the 'baseline'[1] of PubMed and MEDLINE. PubMed is a search engine that provides free access to MEDLINE (and links to full-text articles when possible), NLM's database of citations and abstracts in the fields of medicine, nursing, dentistry, veterinary medicine, health care systems, and pre-clinical

[1]For more information see: `https://www.nlm.nih.gov/bsd/licensee/baseline.html`

| | |
|---|---|
| **Question** | Which protein has been found to interact with phospholamban (PLN) and is also an anti-apoptotic protein? |
| **Snippets** | To identify additional proteins that may interact with PLN, we used the yeast-two-hybrid system to screen an adult human cardiac cDNA library. HS-1 associated protein X-1 (HAX-1) was identified as a PLN-binding partner.<br>Analysis of the anti-apoptotic function of (HAX-1) revealed that the presence of PLN enhanced the (HAX-1) protective effects from hypoxia/reoxygenation-induced cell death. These findings suggest a possible link between the Ca(2+) handling by the sarcoplasmic reticulum and cell survival mediated by the PLN/(HAX-1) interaction.<br>The discovery of the PLN/(HAX-1) interaction therefore unveils an important new link between Ca(2+) homeostasis and cell survival, with significant therapeutic potential. |
| **Answers** | 'The HS-1 associated protein X-1', '(HAX-1)' |

Table 3.1 A training instance of the BIOASQ dataset for Phase B Task B. A human expert poses a question and annotates the titles and abstracts of the biomedical articles by detecting the answer span in the text. In this example, the question is factoid, and the answers that should be extracted from the snippets are 'The HS-1 associated protein X-1' and '(HAX-1)'.

sciences. PubMed publishes yearly a snapshot of MEDLINE/PubMed data in XML format referred to as 'baseline'. NLM releases new and updated XML records on a daily basis. The baseline of 2017 included 24,748,457 publications, while 16,806,115 of them contained only the text of the abstract. By 2022 this number has risen to 33,848,661 publications, while 22,889,477 of them contain only the text of the abstract. [2] Unlike BIOASQ, however, PUBMED baseline does not include questions and answers.

The Medical Question Answering Dataset (MEDQUAD) [18] consists of 47,457 question-answer pairs collected from 12 different NIH[3] websites. The questions cover 37 different topics related to diseases, drugs, and other medical entities such as tests. Contrary to BIOASQ, however, MEDQUAD questions were created using 36 handcrafted patterns (e.g. 'question:When to contact a medical professional about DISEASE?') instead of using questions provided by humans.

PubMedQA[74], is a biomedical question answering (QA) dataset collected from PUBMED abstracts. The task of PubMedQA is to answer research questions by selecting yes/no/maybe

[2]Detailed yearly statistics can be found in: `https://www.nlm.nih.gov/bsd/licensee/baselinestats.html`

[3]The National Institutes of Health (NIH) is a part of the United States Department of Health and Human Services (HHS) and is the primary agency of the U.S. government responsible for biomedical and public health research.

as an answer. Even though PubMedQA contains thousands of QA examples, only $1k$ of these examples are human-authored and $211.3k$ of these examples are artificially generated.

CliCR [180] is a dataset for machine reading comprehension in the medical domain that uses clinical case reports and contains around $100,000$ gap-filling queries about these cases. Almost $12K$ reports were collected from BMJ Case Reports [4] and span the years 2005–2016. A case report is a comprehensive account of a patient's clinical experience that emphasizes uncommon diseases, atypical symptoms of common ailments, and innovative treatment techniques. Each report includes a section called 'Learning Points' which highlights the essential information from the report. These learning points are usually paraphrased fragments of the text and may not exactly match the sentences in the report. To generate questions, we mask a medical entity by using these learning points.

The authors of the paper apply several baselines and neural network readers to the dataset and observe a significant gap in performance (20% F1) between the best human and machine readers. They analyze the skills required for successful answering and show how reader performance varies depending on the applicable skills. They suggest that representing background knowledge by inducing embeddings for entities or integrating ontological knowledge into the models is a promising avenue for future research.

MedQA-USMLE [73] is a dataset that can be used to train and evaluate machine reading comprehension models in the biomedical domain, with a particular focus on diagnostic questions. The dataset contains over $200,000$ questions, along with their corresponding answers, and covers a wide range of diseases, symptoms, and diagnostic criteria. The authors also provide an analysis of the dataset, including the distribution of questions by disease and the difficulty level of the questions. They trained and evaluated several deep learning MRC models but none of them achieved good performance on these data. By publishing their new dataset they encourage future research on biomedical MRC.

MedMCQA [137] is a large-scale, multiple-choice QA dataset designed to address real-world medical entrance exam questions. The dataset contains more than $194k$ high-quality questions covering $2.4k$ healthcare topics and 21 medical subjects. The questions are created using text from entrance exams from the All India Institute of Medical Sciences and the National Eligibility cum Entrance Test (NEET) which is a national-level examination that is conducted by the National Testing Agency (NTA) in India. The dataset is designed to test the reasoning abilities of a model across a wide range of medical subjects and topics, requiring a deeper language understanding. Each sample contains a question, correct answer(s), and other options. The paper provides a detailed explanation of the solution and shows that the dataset is challenging for strong and domain-specific deep learning models, with the best

[4] `https://casereports.bmj.com/`

baseline achieving only 47% accuracy. The authors believe that this dataset would facilitate future research in this direction.

| | |
|---|---|
| **Context** | Chronic urethral obstruction because of urinary calculi, prostatic hyperophy, tumors, normal pregnancy, tumors, uterine prolapse or functional disorders cause hydronephrosis which by definition is used to describe dilatation of renal pelvis and calculus associated with progressive atrophy of the kidney due to obstruction to the outflow of urine Refer Robbins 7yh/9,1012,9/e. P950 . |
| **Question** | Chronic urethral obstruction due to benign prismatic hyperplasia can lead to the following change in kidney parenchyma. |
| **Choices** | Atrophy, Hyperophy, Hyperplasia, Dyplasia |

Table 3.2 An example of a multiple choice question of the MedMCQA dataset. Given a question, the system must select one of the given choices based on the provided context. In this example 'Atrophy' is the answer and 'Dyplasia' cannot be found in the given context which suggests that MedMCQA also includes questions that could only be answered using prior knowledge.

EmrQA [138] describes an approach to generating large-scale QA datasets for specific domains by reusing existing annotations from other NLP tasks. The authors demonstrate this by creating a QA dataset for electronic medical records using annotations from a community-shared dataset, resulting in a corpus called emrQA with $400,000+$ question-answer evidence pairs. The authors conclude that this method has the potential to make a significant impact in domains such as medicine, where obtaining manual QA annotations is difficult.

In their paper, Gu et al. [56] challenge the assumption that mixed-domain pre-training is necessary for biomedical NLP tasks and show that domain-specific pre-training can lead to state-of-the-art results across a range of biomedical NLP applications. To facilitate this study, the authors compiled a comprehensive biomedical NLP benchmark called *BLURB*, which includes a diverse set of tasks such as named entity recognition, relation extraction, document classification, and question answering. They also release their state-of-the-art biomedical BERT models and set up a leaderboard based on BLURB, in order to accelerate research in this field.

cMedQA [222] is a text corpus created by harvesting questions and answers from an online Chinese health and wellness community. The paper focuses on the problem of Chinese medical question-answer matching, which is more challenging than open-domain question-answer matching in English due to the combination of its domain-restricted nature and the language-specific features of Chinese. The authors propose a new end-to-end character-level multi-scale convolutional neural framework that uses character embeddings instead of word embeddings to avoid Chinese word segmentation in text pre-processing. Experimental

results on the cMedQA dataset show that the proposed approach outperforms several strong baselines, and achieves an improvement of top-1 accuracy by up to 19

DrugEHRQA [12] is the first QA dataset that contains question-answer pairs from both structured tables and unstructured notes from publicly available Electronic Health Records (EHRs). The dataset contains medication-related queries, with over 70,000 question-answer pairs. The authors developed a simple baseline model for multimodal QA on EHR, using a modality selection network [58]. The paper provides a benchmark dataset for multi-modal QA systems and opens new avenues of research in improving question answering over EHR structured data using context from unstructured clinical data.

MASH-QA [227] is a dataset for question answering in the consumer health domain, where answers may need to be excerpted from multiple, nonconsecutive parts of text spanned across a long document. The authors of the paper propose a new neural architecture called MultiCo, which is able to capture the relevance among multiple answer spans by using a query-based contextualized sentence selection approach for forming the answer to a given question. The model is evaluated on multiple datasets and is shown to outperform pre-existing state-of-the-art QA models by a wide margin. The authors emphasize that this is the first work that introduces the QA setting with multiple discontinuous answer-spans from a long document.

## 3.3 General Domain Reading Comprehension Datasets

The Text Retrieval Conference (TREC) workshop series[5] encourages research in information retrieval and related applications. The TREC Question Answering track [188] was the first large-scale evaluation of domain-independent question-answering systems. Its dataset contains both biomedical and non-biomedical questions. Multiple NIST assessors independently created question interpretations (questions transformed into one or more executable queries each) and judged responses for the set of non-biomedical questions. To judge the set of medical questions, a NIST assessor was given a set of reference answers from a physician and judged system responses based on the reference answers. Each medical question had only one assessor and none of the assessors was a medical expert.

SemEval (Semantic Evaluation)[6] is an ongoing series of evaluations of computational semantic analysis systems. In the SemEval of 2015 the coordinators organized a total of eighteen tasks into five tracks. In task "Answer Selection in Community Question Answering" of the "Text Similarity and Question Answering" track they offered a dataset inviting systems

[5] http://trec.nist.gov
[6] http://alt.qcri.org/semeval2015/

to retrieve snippets relevant to a question (subtask A) and answer YES/NO questions (subtask B). The questions are derived from a community question-answering system, where there are few restrictions, if any, on who can post and who can answer a question.

The Stanford Question Answering Dataset (SQUAD) [154] is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding given passage. These questions are based on the content of the passage and can be answered by reading through the passage. For each question of SQUAD there are one or more gold answers consisting of snippets or phrases extracted from the passages. With 100,000+ question-answer pairs on 500+ articles, SQUAD is one of the largest and most used manually curated Machine Reading Comprehension (MRC) datasets we are aware of. There is also an online scoreboard[7] where the competitive teams can submit and compare their results.

Squad v2 [153] includes several changes and improvements over Squad v1. Unanswerable questions were added to the corpus which are questions that do not have a corresponding answer in the provided context. This allows models to be trained and evaluated on their ability to identify unanswerable questions. More diverse and representative examples are included, such as questions that require reasoning over multiple sentences or paragraphs, or questions that require an understanding of idiomatic language or figurative language. A new annotation process was followed, which involves a larger and more diverse group of annotators, and a more rigorous quality control process to ensure high-quality annotations. Finally, additional evaluation metrics are provided, such as F1-EM (F1 with exact match) to better evaluate the performance of models on this task.

WikiQA [209] is a publicly available dataset for research on open-domain question answering. Similar to Squad V2, it includes questions for which there are no correct answers, enabling researchers to work on answer triggering, a critical component in QA systems. The authors compare several systems on the task of answer sentence selection on both WikiQA and a previous dataset called QASENT. The authors hope that the WIKIQA dataset will enable further research in answer sentence selection in more realistic settings and provide useful baselines.

Yang et al. [210] introduced HotpotQA, a dataset for question answering that includes $113k$ Wikipedia-based question-answer pairs. The authors argue that HotpotQA is a challenging dataset for QA systems since the questions require finding and reasoning over multiple supporting documents to answer, and the questions are diverse and not constrained to any pre-existing knowledge bases or knowledge schemas. HotpotQA also provides sentence-

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

level supporting facts required for reasoning, allowing QA systems to reason with strong supervision and explain the predictions.

In Onishi et al. [135], the authors present a reading comprehension dataset called "Who-did-What" which contains over 200,000 cloze-style questions constructed from the LDC English Gigaword newswire corpus. The questions are formed by deleting a named entity of type 'Person' from the first sentence of an article. An information retrieval system is then used to select passages with high overlap between the passages and the cloze-style question. The QA models are asked to select the named entity that answers the cloze-style question from a list of named entities of type 'Person' found in the passages. Using pre-existing deep learning MRC models, the authors show that this dataset yields a larger gap between human and machine performance than pre-existing CNN and Daily Mail datasets.

Microsoft MAchine Reading Comprehension (MS MARCO) [132] is a large-scale real-world reading comprehension dataset that contains $1,010,916$ questions and $8,841,823$ companion passages extracted from $3,563,535$ web documents. The questions are anonymized search queries issued through the Bing search engine or Cortana and the companion passages are the corresponding results. For each question, crowd-sourced human annotators were asked to generate answers based on the information contained in the retrieved passages. The annotators were allowed to mark a question as unanswerable based on the passages provided and they were strongly encouraged to form answers in complete sentences In total $182,669$ editorially generated answers were created from the retrieved passages.

CBTest dataset [62] contains cloze-style questions created using passages from children's books. Each cloze-style question (Table 3.3) is a sentence that follows the corresponding passage in its book, with a randomly selected common noun, named entity, verb, or preposition of the sentence removed and turned into a slot to be filled in. It contains approximately $687k$ passage-question instances. It was more recently expanded to BookTest [11], which comprises approximately 14 million passage-question instances, by applying the same methodology to a much larger collection of books.

The creators of BookTest dataset [11] built a dataset similar to the CBTEST dataset. They created a large-scale annotated dataset for training machine reading comprehension (MRC) models. The dataset is extracted from $3,555$ copyright-free books, compared to the CBTEST dataset, which was extracted from only 108 books. The creation of the BookTest dataset followed the same procedure used to create the CBTEST dataset, and it contains $14,140,825$ training examples. The dataset was created by detecting named entities and common nouns in sentences, replacing them with gap tags, and using the preceding 20 sentences as the context document. The training, validation, and test sets were generated from non-overlapping sets of books and the models trained on the BookTest corpus can be evaluated on the original

| | |
|---|---|
| **Context** | 1. The dusk , sweet night seemed to soothe her as it always did .<br>2. She leaned her head against the poplar by the gate .<br>3. How long Spencer Morgan had been standing by her she did not know, but when she looked up he was there .<br>4. In the dim light she could see how haggard and hollow-eyed he had grown .<br>5. He had changed almost as much as herself .<br>6. The girl 's first proud impulse was to turn coldly away and leave him .<br>7. But some strange tumult in her heart kept her still .<br>8. What had he come to say ?<br>9. There was a moment 's fateful silence .<br>10. Then Spencer spoke in a muffled voice .<br>11. " I could n't go away without seeing you once more,<br>Estella, to say good-bye .<br>12. Perhaps you wo n't speak to me .<br>13. You must hate me .<br>14. I deserve it . "<br>15. He paused , but she said no word .<br>16. She could not .<br>17. After a space , he went wistfully on .<br>18. " I know you can never forgive me – no girl could .<br>19. I 've behaved like a fool .<br>20. There is n't any excuse to be made for me . |
| **Cloze-Style Question** | 21. I do n't think I could have been in my right senses , XXXXX. |
| **Choices** | Estella, Morgan, dusk, excuse, first, fool, heart, night, space, tumult |

Table 3.3 An example cloze-style question of the CBTest dataset created using a passage from the book titled 'The Martyrdom of Estella'. 21 random sentences were picked fro the text and the entities of the passage were identified. An entity (Estella) is removed from the 21*st* sentence creating a cloze-style question. To answer the cloze-style question any model must select the correct answer (Estella), from the proposed candidates (i.e. the entities found in the text).

CBTEST data. The authors have shown that using more data can improve performance by up to 14.8%, while attempts to improve the model architecture on the same training data have only given gains of up to 2.1%.

The CNN and Daily Mail datasets [61] were produced in a similar manner. They comprise news articles and cloze-style questions constructed by removing words from sentences summarising the articles; they contain approx. 380k and 880k instances, respectively.

# Chapter 4

# Biomedical Document & Snippet Retrieval

## 4.1 Introduction

Question answering (QA) systems that search large document collections [31, 184, 187] typically use pipelines of components operating at gradually finer text granularities. A fully-fledged pipeline includes components that (i) retrieve possibly relevant documents typically using conventional information retrieval (IR); (ii) re-rank the retrieved documents employing a computationally more expensive document ranker; (iii) rank the passages, sentences, or other 'snippets' of the top-ranked documents; and (iv) select spans of the top-ranked snippets as answers (e.g., named entities). Recently, stages (ii)–(iv) are often pipelined neural models, trained individually [57, 68, 116, 139, 212]. Although pipelines are conceptually simple, errors propagate from one component to the next [64], without later components being able to revise earlier decisions. For example, once a document has been assigned a low relevance score, finding a particularly relevant snippet cannot change the document's score.

In this chapter, we examine the effectiveness of deep learning models for document and snippet retrieval. We present pipelined and joint approaches to retrieve both relevant documents and snippets given a biomedical question. Our research has led to three distinctions in the BIOASQ challenges as well as the development of the first joint model ever to tackle document and snippet retrieval in the biomedical domain simultaneously. The best deep learning approach has also been used in real-world scenarios to (i) answer COVID-related questions through an online tool and (ii) assist biomedical experts in relevant literature identification for biomedical systematic reviews.

## 4.2 Related Work

Past work on information retrieval has investigated several techniques, ranging from traditional IR approaches to deep learning models and pipelines. Neural document ranking [57, 68, 139, 69, 116] only recently managed to improve the rankings of conventional IR; see Lin et al. [102] for caveats. Document or passage ranking models based on BERT [42] have also been proposed, with promising results, but most approaches use only simplistic task-specific layers on top of BERT [206, 133], similar to our use of BERT for document scoring (to be discussed in section 4.4.1). An exception is the work of MacAvaney et al. [112], who explored combining ELMO [149] and BERT with complex neural IR models, namely PACRR [68], DRMM [57], KNRM [39], CONVKNRM [197], an approach that we also explore in this thesis by combining BERT with PDRMM in BJPDRMM and JBERT (to be discussed in section 4.5.1). However, we jointly retrieve both documents and snippets, whereas MacAvaney et al. [112] retrieve only documents.

Models that directly retrieve documents by indexing neural document representations, rather than re-ranking documents retrieved by conventional IR, have also been proposed [52, 6, 81], but none addresses both document and snippet retrieval. Yang et al. [204] use BERT to encode, index, and directly retrieve snippets, but do not consider documents. Lee et al. [95] propose a joint model for direct snippet retrieval (and indexing) and answer span selection, again without retrieving documents. However, creating vector representations of millions of snippets and creating an index to store these representations is computationally expensive. For example, in PUBMED (discussed in chapter 3.2 there are more than 30 million documents. Even if only 10 snippets were indexed for each document it would need approximately 1TB to store the index created using vectors of size 768 when a BERT model is used. In contrast, when using a BM25-based retrieval engine only 175GB of storage space is required.

No previous work combined document and snippet retrieval in a joint neural model which is one of the key contributions of this chapter. This may be due to existing datasets, which do not provide both gold documents and gold snippets, with the exception of BIOASQ [184], which is however small by today's standards (2.7k training questions). For example, Pang et al. [139] used much larger clickthrough datasets from a Chinese search engine, as well as datasets from the 2007 and 2008 TREC Million Query tracks [150], but these datasets do not contain gold snippets. SQUAD [154] and SQUAD v.2 [153] provide 100k and 150k questions, respectively, but for each question they require extracting an exact answer span from a single given Wikipedia paragraph; no snippet retrieval is performed, because the relevant (paragraph-sized) snippet is given. Ahmad et al. [5] provide modified versions of SQUAD and Natural Questions [87], suitable for direct snippet retrieval, but do

not consider document retrieval. SearchQA [50] provides 140k questions, along with 50 snippets per question. The web pages where the snippets were extracted from, however, are not included in the dataset, only their URLs, and crawling them may produce different document collections, since the contents of web pages often change, pages are removed etc. MS-MARCO [132] was constructed using 1M queries extracted from Bing's logs. For each question, the dataset includes the snippets returned by the search engine for the top-10 ranked web pages. However, the gold answers to the questions are not spans of particular retrieved snippets, but were freely written by humans after reading the returned snippets. Hence, gold relevant snippets (or sentences) cannot be identified, making this dataset unsuitable for our purposes.

Wang et al. [190] managed to achieve a Mean Average Precision (MAP) score [1] of 0.7134 in the TREC QA dataset [187] achieving at that time a new state of the art. They used a combination of a three-layer stacked bidirectional recurrent neural network (biRNN) [163, 166] with LSTM units [63] (see also section 2.2.1) as well as keyword matching using the BM25 [156] score (see section 2.1). They claim that their experiments provide strong evidence that distributed (embeddings extracted from the RNN) and symbolic (keyword matching) representations encode complementary types of knowledge, which are all helpful in identifying answer sentences.

Severyn et al. [167] describe a new deep learning architecture for re-ranking short texts, such as questions and documents limited to one sentence each. The architecture is based on two distributional sentence models using convolutional neural networks that map input sentences to distributional vectors and learn the semantic similarity between them. The advantages of the proposed model include its ability to learn from unsupervised corpora, use a rich representation of query-document pairs, and ease of adding additional similarity features. The model outperforms previous state-of-the-art systems in answer sentence selection and Microblog retrieval tasks and does not require manual feature engineering or external resources.

The next year, Yang et.al. [202] proposed an attention-based neural matching model for ranking short answer texts based on whether they answer or not the question. They used convolutional layers to extract token contextual embeddings for a sentence and a question. Yang et al. also introduced a new attention mechanism which they called the 'value-shared weighting scheme' and applied it to the dot product similarity between the contextual embeddings of the question and the sentence. Their attention mechanism divides the similarity range $[-1,1]$ into $K$ bins (for example if the size of a bin is set to 0.1 then

[1] `https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision`

20 bins are created). A trainable parameter is assigned to each bin so that pairs with close similarities are attended by the same trainable weight. They experimented with their model using additional features such as word overlap features, IDF weighted word overlap features, and BM25 and they managed to get a MAP score of 0.7495 on the TREC QA dataset using their model and only one additional feature, i.e. the Query Likelihood (QL) [201] score using the toolkit LambdaMART [26].

Suggu et al. [178] presented a novel approach "Deep Feature Fusion Network (DFFN)" for AQP (Answer Quality Prediction) which combines hand-crafted features (HCF) with deep learning features. The DFFN architecture takes in a question, answer, and metadata and predicts the quality of the answer. The approach proposes two different models, DFFN-CNN and DFFN-BLNA, which differ in the way they model the input question-answer pair. DFFN enriches the features learned by CNN and BLNA by incorporating external resources such as Wikipedia, Google Cross-Lingual Dictionary (GCD) [177], and Clickthrough Data [67]. The results show that DFFN outperforms other approaches and achieves state-of-the-art performance on the standard SemEval-2015 and SemEval-2016 benchmark datasets (see section 3.3 for details on the SemEval series).

## 4.3 Data Handling

We use data from PUBMED (see section 3.2), a search engine that provides free access to MEDLINE, NLM's database of citations and abstracts in the fields of medicine, nursing, dentistry, veterinary medicine, health care systems, and pre-clinical sciences. The document collection consists of approx. 30M 'articles' (titles and abstracts only) from the 'MEDLINE/PubMed Baseline' collection (years 1900-2020).[2] We discarded the approx. 10M articles that contained only titles, since very few of these can be used for Information Retrieval or Question Answering. For the remaining 20M articles[3], a document was constructed from the concatenation of each title and abstract. These documents were then indexed using Galago[4] and ElasticSearch[5], removing stop words and applying Krovetz's stemmer [84].

[2]Available from `https://www.nlm.nih.gov/databases/download/pubmed_medline.html`.

[3]By January 2023 this number rose to 24M articles.

[4]We used Galago version 3.10. Consult `http://www.lemurproject.org/galago.php`.

[5]We used ElasticSearch 5.0.1. Consult `https://www.elastic.co/guide/en/elasticsearch/reference/5.0/index.html`.

### 4.3.1 Biomedical Word Vector Representations

Word embeddings were pre-trained by applying WORD2VEC [119] to the 28M 'articles' (years 1900-2018) of the MEDLINE/PubMed collection. IDF values (see section 2.1) were computed over the 18M articles that contained both titles and abstracts. We used the GenSim implementation of WORD2VEC (skip-gram model), with negative sampling, and window size set to 5, to produce word embeddings of 200 dimensions.[6] For tokenization, we used the 'bioclean' tool provided by BIOASQ.[7] In snippet retrieval, we used NLTK's English sentence splitter.[8]

## 4.4 Pipelined Methods for Document and Snippet Retrieval

We first describe pipeline approaches for document and snippet retrieval. These pipelines, achieved state of the art scores compared to other pipeline models but also act as strong baselines against the proposed joint models.

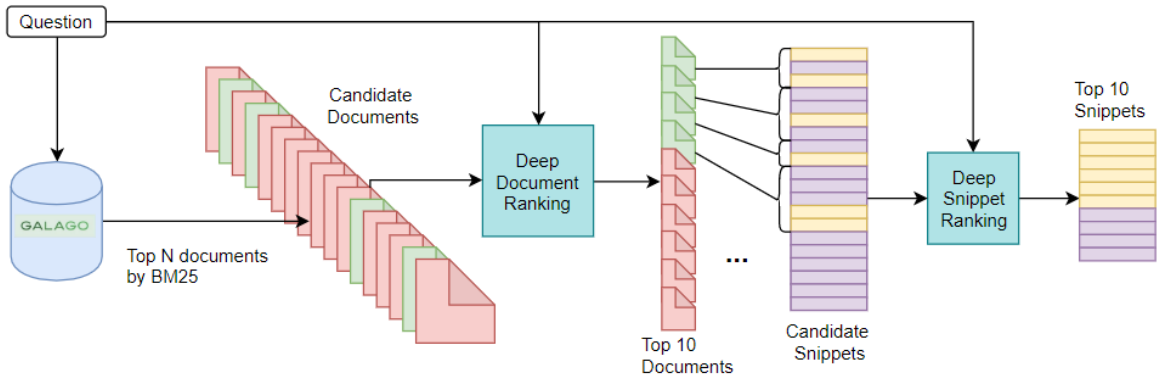


Fig. 4.1 Architecture of our pipelined document and snippet retrieval systems. The IR engine retrieves candidate relevant documents (left). A neural document retrieval model ranks the retrieved documents and returns the top 10. Then a neural snippet retrieval model ranks the snippets from the 10 documents and returns the top 10 snippets.

[6] Consult `https://radimrehurek.com/gensim/models/word2vec.html`. All other hyper-parameters where set to default values. We used Gensim v. 3.3.0. The word embeddings and code of our experiments are available at `https://github.com/nlpaueb/aueb-bioasq6`.

[7] The tool accompanies an older set of embeddings provided by BIOASQ. See `http://participants-area.bioasq.org/tools/BioASQword2vec/`.

[8] We used NLTK v3.2.3. See `https://www.nltk.org/api/nltk.tokenize.html`.

### 4.4.1 Document retrieval in pipelined methods

#### BM25

In document retrieval, we re-rank documents retrieved by a traditional IR engine using BM25 (see section 2.1). As a first approach, we examine the rank of documents based on BM25 scoring without re-ranking the documents. This way we can measure the improvement of each of the following re-ranking methods. During retrieval, the user poses a question in natural language. Stopwords are removed from the question and the remaining words are submitted as a query to the IR engine. The IR engine sorts the documents in descending order according to their BM25 score and the top $N$ documents are selected.[9]

#### TERM-PACRR

The first model we use for document retrieval is TERM-PACRR [24, 116], a modification of PACRR [68].[10] To train TERM-PACRR, we use mini-batches containing randomly selected relevant and irrelevant documents (in equal numbers) from the top $N$ documents that the IR engine retrieves per training question, and we minimize binary cross-entropy (Equation 4.3) for each document in the mini-batch. As in [24], we use a final linear layer that combines the TERM-PACRR score with extra features:

- the number of common tokens in the question and the sentence,
- the number of common token bigrams in the question and the sentence,
- the sum of the IDF scores of the common tokens of the question and sentence,
- the BM25 score of the document as computed by the retrieval engine,

The overlap and IDF-weighted overlap scores are computed using Equation 4.1 where $Q$ represents the query terms, $D$ represents the document terms, and $\text{IDF}(t)$ is the inverse document frequency for term $t$. In the case of the bigram overlap instead of query terms and document terms, $Q$ and $D$ represent bigrams found in the query and document respectively.

$$overlap(Q,D) = \frac{|Q \cap D|}{min(|Q|,|D|))}$$
$$\text{IDF-weighted-overlap}(Q,D) = \sum_{t \in Q \cap D} \text{IDF}(t) \tag{4.1}$$

[9] In our experiments we set $N = 100$

[10] TERM-PACRR is called TERM-DRMM in [116].

#### PDRMM

**PO**oled **SI**milariTy **D**eep **R**elevance **M**atching **M**odel (PDRMM) is our first deep learning model for document re-ranking (Figure4.2). Given a query $q = \langle q_1 ... q_n \rangle$ of $n$ query terms (q-terms) and a document $d = \langle d_1 ... d_m \rangle$ of $m$ terms (d-terms), PDRMM computes context-sensitive term embeddings $c(q_i)$ and $c(d_i)$ from the static WORD2VEC embeddings $e(q_i)$ and $e(d_i)$ by applying two stacked convolutional layers with trigram filters, residuals, and zero padding to $q$ and $d$, respectively. PDRMM then computes three similarity matrices $S_1, S_2, S_3$, each of dimensions $n \times m$ (Figure 4.2 upper left part). Each element $s_{i,j}$ of $S_1$ is the cosine similarity between $c(q_i)$ and $c(d_j)$. $S_2$ is similar, but uses the static WORD2VEC embeddings $e(q_i), e(q_j)$. $S_3$ uses One-Hot vectors for $q_i, d_j$, signaling exact matches. A One-Hot vector representation of a term is a vector of size equal to the number of unique words in a corpus (vocabulary), where each dimension in the vector represents a unique word in the vocabulary. Each dimension of the vector will be a binary value of either 0 or 1. The value is set to 1 at the index of the word in the vocabulary, and 0 at all other dimensions. To each matrix ($S_1$, $S_2$, or $S_3$) we apply three row-wise pooling operators to extract 9 features for each q-term: max-pooling, average-pooling, and the average of $k$-max. Max-pooling is used to obtain the similarity of the best match between the q-term of the row and any of the d-terms. Average pooling is used to obtain the average match of each q-term to all d-terms. The average of $k$-max pooling is used to obtain the average similarity of the $k$ best matches per q-term.[11]

We concatenate the three features extracted from each row of the three similarity matrices (9 features in total) and concatenate them to obtain a new matrix $S'$ of dimensions $n \times 9$ (Figure 4.2, right). Each row of $S'$ indicates the similarity of the corresponding q-term to any of the d-terms, through three different views of the terms (One-Hot, static, context-aware embeddings). Each row of $S'$ is then passed to a Multi-Layer Perceptron (MLP) to obtain a single match score per q-term. This MLP consists of one dense layer with 8 neurons and leaky RELU activation function, followed by a second dense layer with 1 output and no activation function.

Each context-aware q-term embedding is also concatenated with the corresponding IDF score (Fig. 4.2, bottom left) and passed to another linear layer that computes a score for each q-term. A SoftMax activation function is then applied across all the q-term scores to compute the importance of each q-term (e.g., words with low IDFs may not be helpful to answer the question).[12] Let $v$ be the vector containing the $n$ match scores of the q-terms, and $u$ the vector of the corresponding $n$ importance scores (Fig. 4.2, bottom).

[11] In our experiments, $k = 5$. We added the average pooling to PDRMM to balance the other two pooling operators that favor long documents.

[12] The importance scores of the q-terms can also be viewed as self-attention scores.

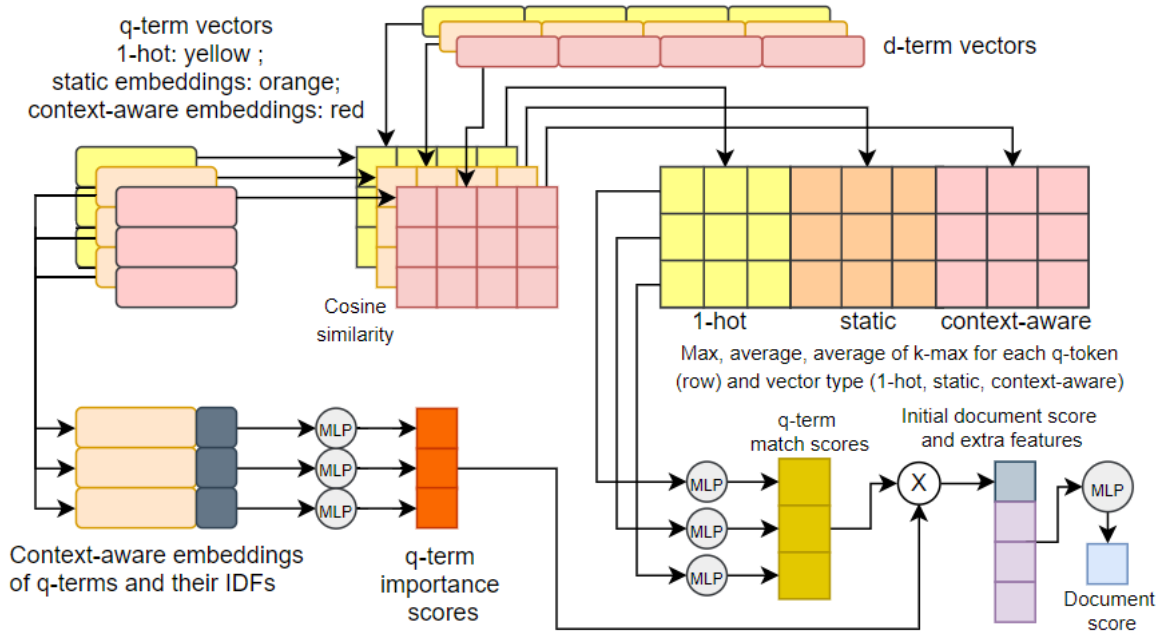


Fig. 4.2 PDRMM for *document* scoring. The same model (with different trained parameters) also scores *sentences* in the PDRMM+PDRMM pipeline and the joint JPDRMM model (adding the layers of Figure 4.9).

We extract an initial relevance score for the document as $\hat{r}(q,d) = v^T u$, which is then concatenated with the extra features (The same features used in TERM-PACRR). An MLP computes the final relevance $r(q,d)$ from the five features. [13] PDRMM is trained on triples $\langle q,d,d'\rangle$, where $d$ is a relevant document from the top $N$ that the IR engine returned for question $q$, and $d'$ a randomly sampled irrelevant document among the top $N$. Hinge loss (Equation 4.2) is used, requiring $r(q,d)$ to exceed $r(q,d')$ by a margin.[14]

$$Hinge\ Loss(q,d,d') = max(0, margin + r(q,d') - r(q,d)) \tag{4.2}$$

#### BERT

As our second model for document retrieval, we use a BERT-based model. When fine-tuning BERT on BIOASQ data or when using it at test time, we feed it with the concatenation of a question and a (relevant or irrelevant) document. As standard, a special [CLS] token is added to the start of the concatenation, while a [SEP] token separates the question from the document (concatenated title and abstract), as illustrated in Figure 4.3. The output vector

[13] This MLP also consists of one dense layer with 8 neurons and leaky RELU activation function, followed by a second dense layer with no activation function.

[14] In our experiments we set margin to 1.0

of BERT for the [CLS] token is passed through a logistic regression layer (linear layer with sigmoid) to obtain a BERT-based score for the document. This score is then concatenated to extra features of the document (BM25 score and string overlap features), which are the same as in TERM-PACRR. Finally, another logistic regression layer is applied to the concatenated vector to get the final score of the document.

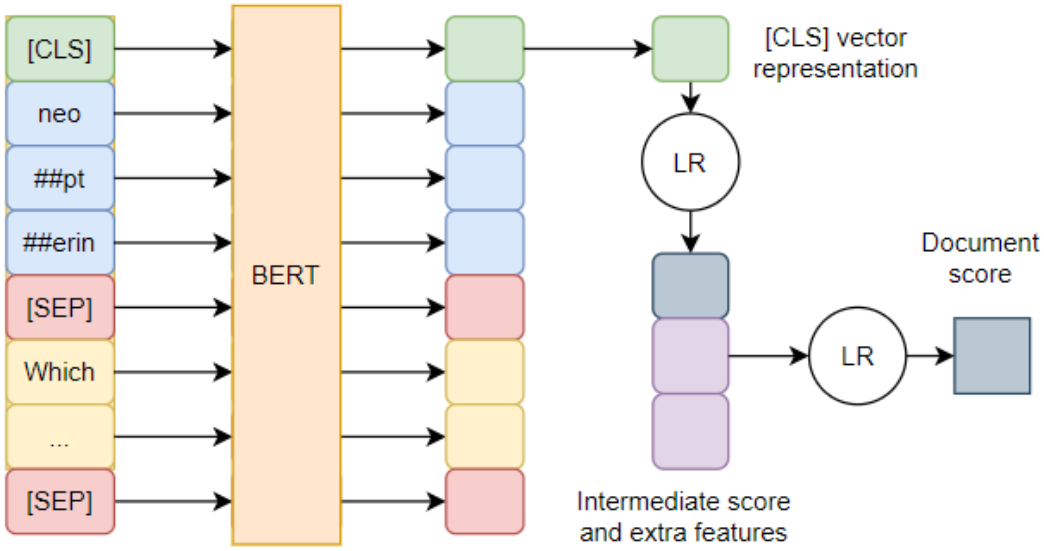


Fig. 4.3 *Document* scoring with BERT.

During fine-tuning, negative samples (irrelevant documents to be concatenated with a training question) are drawn randomly from the non-relevant (according to the expert annotators) documents in the list of top $N$ documents that the conventional IR system (see BM25 in section 2.1) returned for the particular question. Critically, we found that using two losses per training instance helped accuracy. The first loss is the binary cross-entropy of the final document score (Figure 4.3 last gray box). The second loss is also binary cross-entropy but computed on the BERT-based score, before concatenating it with the extra features (Figure 4.3 first gray box). The two losses are summed.

Similarly to [24], we also experiment with a *high-confidence* version of BERT.[15] In this model, only documents with scores (probabilities of being relevant) greater than 0.01 were returned as relevant, hence fewer than 10 documents (the maximum allowed in BIOASQ) might be returned. This helped to improve the snippet retrieval component which was now focused only on the most relevant documents.

[15] In [24], the high-confidence models were for another model, ABEL-DRMM.

Binary Binary cross-entropy loss can be defined as:

$$\mathscr{L}(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)]$$

where:

N is the number of samples,

$y_i$ is the true label for the $i-th$ sample,

$\hat{y}_i$ is the predicted probability for the $i-th$ sample to be positive and,

$\mathscr{L}(y,\hat{y})$ is the binary cross-entropy loss for the prediction and ground truth pairs. (4.3)

### 4.4.2 Snippet retrieval in pipelined methods

For snippet retrieval in pipelined methods, we trained state-of-the-art models to also rank snippets of texts from the retrieved documents. For simplicity, we examine the sentences of the top-ranked documents as snippets.

As in document retrieval, we use additional hand-crafted features:

1. the length of the question in characters,
2. the length of the sentence in characters,
3. the number of common tokens in the question and sentence,
4. the number of common tokens in the question and sentence excluding stopwords,
5. the number of common bigrams in the question and sentence,
6. the BM25 score of the sentence computed using all retrieved sentences as our corpus[16],
7. the BM25 score of the document as computed by the retrieval engine,
8. the sum of the IDF scores of the common tokens of the question and sentence,
9. the sum of the IDF scores of the common tokens of the question and sentence, excluding stopwords,
10. the sum of the IDF scores of the common tokens of the question and sentence, normalized by the sum of the question's tokens' IDF scores.

[16]Instead of using all sentences of the 20M documents found in PUBMED we use only the sentences of the retrieved documents as our corpus and compute the BM25 score (Equation 2.1).

### BM25

Similarly to document retrieval we use BM25 to also rank the sentences of the top-ranked documents. A BM25 score (section 2.1) is computed using the user's question and each sentence separately (out of the context of the document).

#### BCNN

We apply two established deep learning models for sentence pair matching namely Basic Bi-CNN (BCNN) and Attention-Based Convolutional Neural Network for Modeling Sentence Pairs (ABCNN3) [212]. We re-implemented BCNN and ABCNN3 to extract a matching score for each question-document pair.

For BCNN, given a question of $Q$ terms and a snippet of $S$ terms, we compute three similarity scores $sim_1$, $sim_2$ and $sim_3$ (Figure 4.4, dark pink boxes on the right) between the question and the snippet. To compute $sim_1$ we collect the WORD2VEC embeddings for both question and snippet terms (Figure 4.4, left part) and apply global average pooling to get one vector representation for the question and one for the snippet. Then $sim_1$ is computed as the cosine similarity of the two vectors.

We pad the embedding vectors (gray boxes in Figure 4.4) and apply a 4-gram convolutional layer with a Tanh activation function on the padded embeddings. Then windowed average pooling 4.4 is applied to the result of the convolution to get contextual vector representations for the terms (Figure 4.4, middle). To compute $sim_2$ we apply global average pooling to the contextual vectors of the question and the snippet (Figure 4.4 middle top and bottom) followed by cosine similarity.

Windowed average pooling is defined as:

$$Out(i,j) = \frac{\sum_{k=0}^{w} In(i+k,j)}{w}$$

where:

$$\begin{aligned} &In \text{ is the input matrix of size } K \times L, \\ &Out \text{ is the output matrix of size } (K-w+1) \times L, \\ &w \text{ is the size of the window (we set } w = 4), \\ &i \text{ is the row of the element we compute in the output matrix,} \\ &j \text{ is the column of the element we compute in the output matrix.} \end{aligned} \tag{4.4}$$

A second 4-gram convolutional layer with a Tanh activation function is applied followed by global average pooling[17] to get one vector that intuitively embeds the meaning of the entire sequence. The entire process is applied to both the $Q$ terms of the question and the $S$ terms of the snippet and two vectors are computed (Figure 4.4, right). $sim_3$ is computed as the cosine similarity between the two vector representations.

The three cosine similarity scores ($sim_1$, $sim_2$, and $sim_3$) are concatenated to extra hand-crafted features (Figure 4.4, pink boxes on the right). A Logistic regression layer consumes the concatenated vector and computes a relevance score between the snippet and the question.

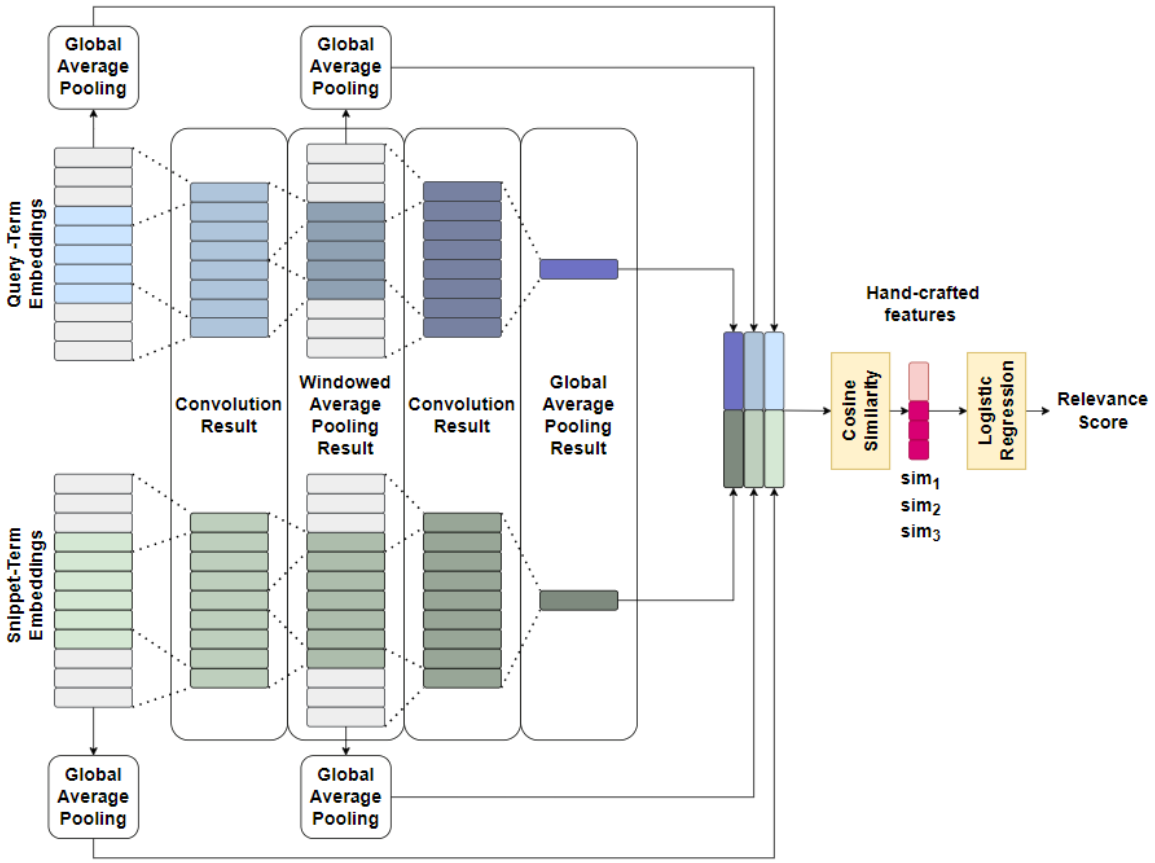


Fig. 4.4 BCNN [212] scoring snippets relative to a question. The example illustrates a question of 5 terms, a snippet of 7 terms, and a single convolution filter of width $w = 3$. Zero-padding is shown as empty gray boxes. In each convolution/pooling block, the convolution layer is followed by a windowed-average pooling of the same width $w$ to preserve the dimensionality of the input to the block. Thus convolution/pooling blocks can be repeated, making the model arbitrarily deep.

[17] This is similar to setting the window size in window average pooling equal to the length of the input sequence.

#### ABCNN3

Attention-Based Convolutional Neural Network for Modeling Sentence Pairs (ABCNN3) resembles the BCNN model. The difference lies in the attention mechanisms we apply before and after each convolution. Given a question of $Q$ terms and a sentence of $S$ terms ABCNN3 computes three similarity scores $sim_1$, $sim_2$ and $sim_3$ between the question and the sentence. We collect the WORD2VEC vector representations of size $E$ for the question terms and the sentence terms. In order to compute $sim_1$, global average pooling is used (see also BCNN) on the vector representation of the question's terms and the vector representation of the sentence's terms (Figure 4.5, part 1, top section), to compute one vector representation for the entire question and a second vector representation for the sentence. Then $sim_1$ is computed using cosine similarity between the two vectors.

To compute $sim_2$ the two sequences of embeddings are padded to the size of the longest sequence (denoted as $L$ in Figure 4.5) in the corpus. A euclidean distance matrix $ED1$ of size $L \times L$ is computed using Equation 4.5 between every embedding of the question and every embedding of the sentence. The intuition behind the similarity matrix $ED1$ is that it can capture similarities between the question and the sentence which can be then used to attribute higher attention values to similar tokens of the question and the sentence.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{4.5}$$

A trainable attention matrix $AW$ of size $L \times E$ is multiplied by the matrix that holds the euclidean distance scores ($ED1$) to compute attention scores on the sentence terms (Figure 4.5, part 1, middle). A second matrix multiplication is applied between the trainable attention matrix $AW$ and the transposed matrix of the euclidean distances ($ED1^T$) to compute attention scores on the question terms. The attention matrices are stacked to the input vector representations of the question and the sentence (bottom section of part 1 in Figure 4.5) and a convolutional layer is applied on the stacked matrices (top section of part 2 in Figure 4.5).

A second similarity matrix ($ED2$) is computed using again the euclidean distance between the contextual vector representations of the question and the contextual vector representations of the sentence computed by the convolution. Then a column-wise summation and a row-wise summation are computed on the similarities to capture one score per token as the total similarity on the question terms (Figure 4.5, part 2, middle, 'row sum') and the sentence terms (Figure 4.5, part 2, middle, 'column sum').

The summation results are expanded to match the size of the contextual vector representations and the two matrices are staked so that the total similarity score of a token is replicated $E$ times and aligned with all values of the contextual vector representation of the token. Sum

pooling is used to extract the final contextual vector representations of the sentence and the question terms. A global average pooling mechanism computes a final vector representation for the entire question and another one for the entire sentence. Finally, the $sim_2$ score is computed as the cosine similarity of the pooled vectors (Figure 4.5, part 2, red box at the bottom).

To compute $sim_3$ the same process that computes $sim_2$ is followed. This time however instead of using the WORD2VEC embeddings of the question and the snippet as input, the contextual embeddings computed in the former steps (Figure 4.5, part 2, 'Sum Pooling results') are used. The three similarity scores computed by ABCNN3 ($sim_1$, $sim_2$, $sim_3$) are concatenated along with the extra hand-crafted features (just like when using the BCNN model), and a final dense layer with SoftMax activation function decides whether the sentence is relevant to the question or not.

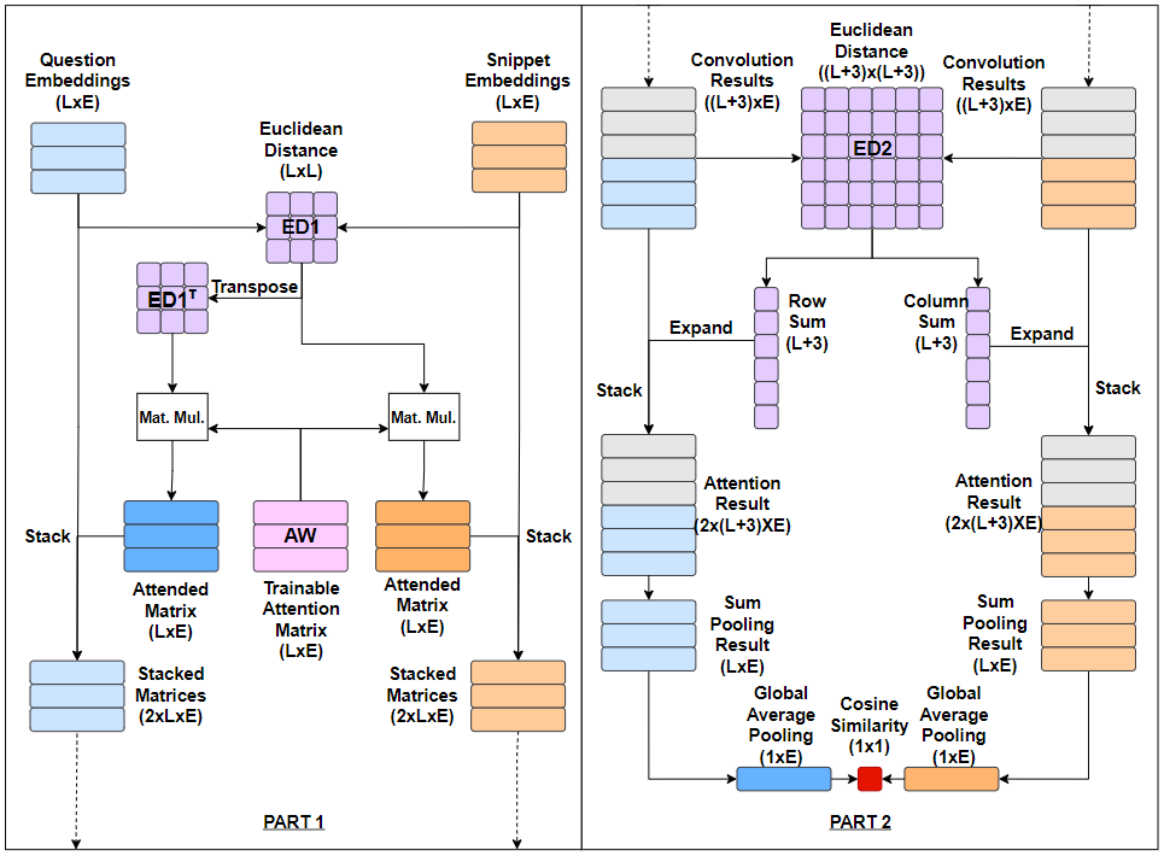


Fig. 4.5 Attention mechanisms in the ABCNN3 model. ABCNN3 combines the two attention mechanisms in order to improve the performance of BCNN. Part 2 is a continuation of part 1.

#### PDRMM

We use the same process as in document retrieval (see Figure 4.2) to extract a score for each sentence and then concatenate that score to the external features of the sentence. In this case, instead of using the entire document and the question as input to the PDRMM model, we use the query and a sentence instead. Given that we computed a vector that contains the sentence score and the extra features for each sentence, we use a convolutional layer (Figure 4.6) to extract each sentence's final score, combining the sentence's vector and the surrounding sentences' vectors.

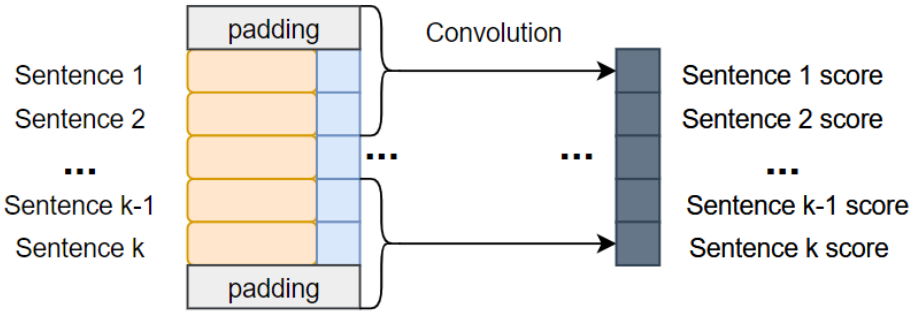


Fig. 4.6 Convolution on Sentence PDRMM. The extra features (orange boxes) are concatenated with the sentence scores. A 3-gram convolution learns to combine the extra features and the scores of three sentences and computes a final score for each sentence.

#### BCNN-PDRMM

We use the same process as when using PDRMM for document retrieval (Figure 4.2), to extract the intermediate scores of the sentences. Using BCNN, we apply average pooling and cosine similarity to the input embeddings as well as to the output of each convolutional layer extracting three scores for each sentence ($sim_1$, $sim_2$, and $sim_3$ in Figure 4.4). We concatenate the PDRMM intermediate score and the BCNN scores with the external features and apply the convolution just like PDRMM to extract the final scores of the sentences.

#### ABCNN3-PDRMM

In the original PDRMM model (see also section 4.4.1) three pooling mechanisms are used to compute similarities between the terms of the question and the terms of a document (see 4.2 right part). In ABCNN3-PDRMM we use the same pooling mechanisms but instead of using a convolutional layer to extract contextual vector representations for the terms we use

the output of the convolutional layers found in ABCNN3 (Figure 4.5 bottom right section 'Attention result').

#### BERT

We use the same architecture as when using BERT for scoring document relevance with respect to a question (section 4.4.1). However, instead of using the concatenation of the document and the question as input, this time we use the concatenation of the snippet and the question. Additionally, instead of using four hand-crafted features as in BERT for document retrieval, now we use the hand-crafted features used in all sentence retrieval models (10 features in total).

## 4.5 Joint Methods

We propose novel models for joint document and sentence retrieval. Our models compete with state-of-the-art systems in both tasks using far fewer trainable parameters than other proposed models. To our knowledge, we introduced the first model ever jointly trained for documents and sentences. Our jointly trained models managed to surpass all the competitive models in BIOASQ-7 and BIOASQ-8 challenges.

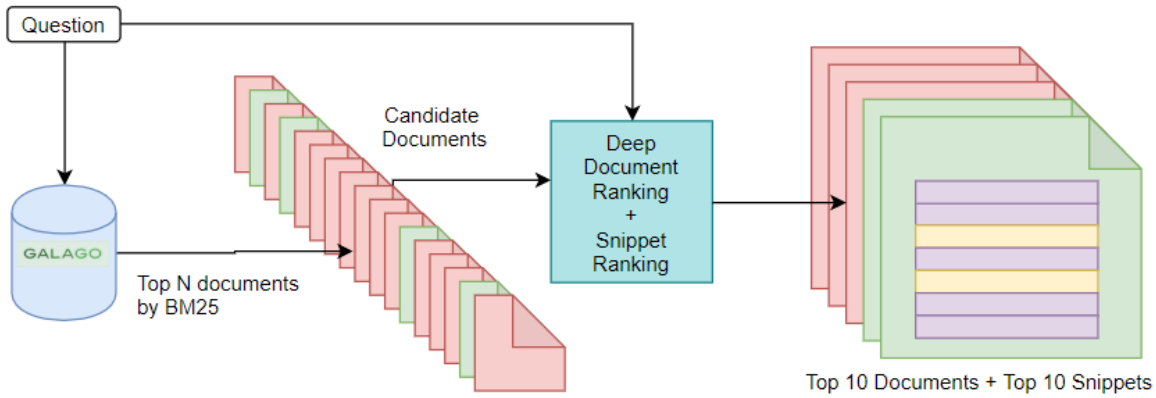


Fig. 4.7 The architecture of our joint document and snippet retrieval systems. The IR engine retrieves candidate relevant documents. The joint neural model assigns scores to the sentences of the retrieved documents and their snippets. We return the 10 documents with the highest scores and the 10 snippets of those documents with the highest scores. In BIOASQ 7 we participated with systems that use Galago as our IR engine. In BIOASQ 8 we replace Galago with the ElasticSearch engine.

### 4.5.1 JPDRMM, BJPDRMM and GRAPH-JPDRMM

We create a joint PDRMM-based model, called JPDRMM, which given a question and a document, outputs relevance scores for each sentence (snippet) of the document, along with a relevance score for the entire document. JPDRMM applies the same process described in section 4.4.2 to compute a score for each sentence in the document. The maximum score of all the sentences is selected and concatenated to the extra features of the document (left part of Figure 4.9), which are the same as when PDRMM scores documents (see section 4.4.1). We have also experimented with other pooling operators to obtain the document score from the sentence scores, including combinations of max-pooling, average pooling, and the average of top $k$ pooling, but they did not improve performance.

The score of the document is computed by applying an MLP to the concatenated features. The MLP consists of one dense layer with 8 neurons and leaky RELU activation function, followed by a second dense layer with no activation function. The scores of the sentences are then revised to take into account the score of the entire document; the intuition is that snippets from relevant documents are more likely to be relevant. To do so, we concatenate the score of each sentence to the document score (Figure 4.9, right part), and pass each pair of sentence-document scores through a logistic regression layer to obtain the final sentence score.

Like the original PDRMM, JPDRMM is trained on triples $\langle q, d, d' \rangle$, where $q$ is a question, and $d, d'$ are relevant and irrelevant documents, respectively, sampled from the top $N$ documents returned by the IR engine for $q$. In this case, however, we apply a sentence-splitter to $d$ and $d'$, and use JPDRMM to obtain relevance scores for $d$, $d'$, and each one of their sentences. We compute a document hinge loss from the scores of $d$ and $d'$ as when PDRMM scores documents, and a binary cross-entropy (Equation 4.3) loss for each sentence (relevant or irrelevant) of $d$ and $d'$ as when PDRMM scores sentences. The document hinge loss is added to the average sentence cross-entropy loss (averaged over all the sentences of $d$ and $d'$), and their sum is used to train the entire model via back-propagation.

We create three versions of JPDRMM: one using pre-trained WORD2VEC embeddings, one using pre-trained word embeddings using graph link prediction [82], and one using pre-trained embeddings obtained from the top layer of the publicly available BERT BASE instance [42].[18] We call W2V-JPDRMM, GRAPH-JPDRMM, and BJPDRMM the three versions, respectively.

To train word vector representations used in GRAPH-JPDRMM we had to construct a biomedical graph that contained thousands of biomedical entities so that out-of-vocabulary words would be rare and the embeddings could cover a wide range of biomedical terms. The

[18] We also experimented with BIOBERT [94], but there was no notable improvement.

node embedding method we used is an extension of NODE2VEC [55] that considers both the topology of the graph it is applied to and the text associated with each node of the graph. In our case, nodes are biomedical entities and the text of each node is the (often multi-word) English name of the corresponding entity. The graph node embedding method uses an RNN to obtain a node embedding from the word embeddings of the text (name) of the node and then applies graph convolutions to make sure that the embeddings of nodes with common neighbors are close to each other. In effect, the embeddings of two nodes (entities) end up being close to each other if the two nodes have similar names (e.g., 'acute cardiomyopathy', 'cardiomyopathy') or similar neighbors.

To construct the entity co-occurrence graph, we used PUBTATOR [193] to identify the biomedical entities in a randomly selected set of approx. 5 million PUBMED abstracts. Whenever a biomedical entity was found in the same abstract as another one, a link between the two entities was added to the graph. We then pruned links corresponding to co-occurrences with frequencies lower than 10. Although the graph embedding method is primarily intended to generate node embeddings, it also generates word embeddings, which we used as an alternative to WORD2VEC embeddings. The intuition was that nodes (entities) with similar neighborhoods in the co-occurrence graph are probably related, the graph node embedding method places their embeddings close to each other, and this might also help place close to each other the embeddings of the words of the names of the two related nodes since node embeddings are based on the word embeddings of the node names. We call GRAPH-JPDRMM the JPDRMM version that uses word embeddings obtained via the graph embedding method, and W2V-JPDRMM the original JPDRMM version with biomedical WORD2VEC embeddings.

BERT's tokenizer splits words into subword units (wordpieces) [195]. In BJPDRMM, in order to use IDF scores of entire words and compute exact matches across entire words, as in W2V-JPDRMM, we reconstruct the words from the subword units before feeding them to the rest of the model. Also, we use BERT's top-level embedding for the first wordpiece of each reconstructed word as the pre-trained embedding of that word.

In BJPDRMM we also experimented with finetuning the BERT model when training JPDRMM. We call BJPDRMM-NF the model where the BERT part is not fine-tuned. In another variant of BJPDRMM, called BJPDRMM-ADAPT, the input embedding of each token is a linear combination of all the embeddings that BERT produces for that token at its different Transformer layers. The weights of the linear combination are learned via back-propagation. This allows BJPDRMM-ADAPT to learn which BERT layers it should mostly rely on when obtaining token embeddings. Previous work has reported that representations from different BERT layers may be more appropriate for different tasks [158]. BJPDRMM-ADAPT-NF

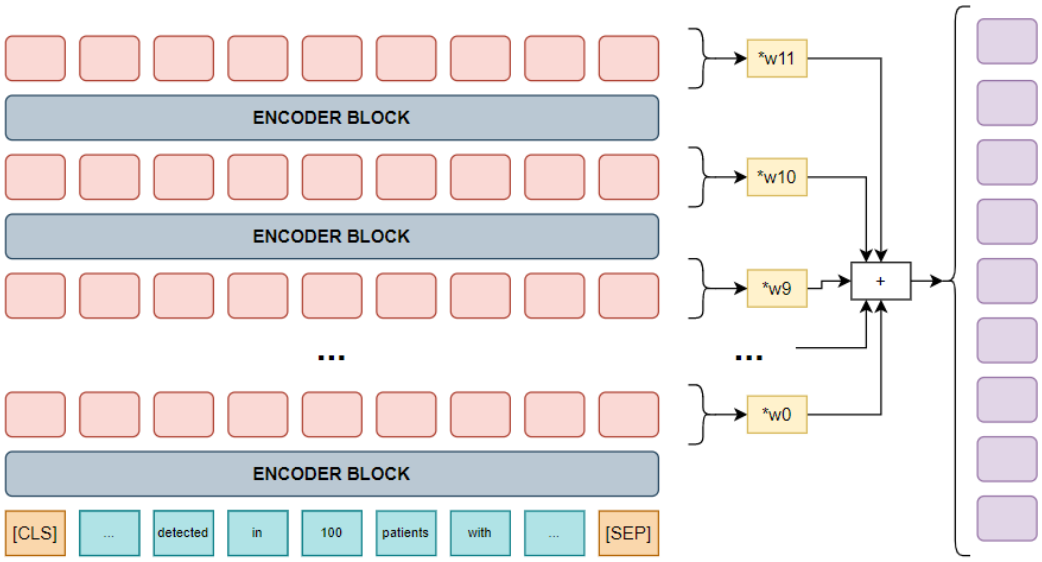


Fig. 4.8 The BJPDRMM-ADAPT model combines the vector representations computed by each encoder block by weighting the vectors of each layer by a trainable scalar. Each vector representation computed in layer $i$ is multiplied by a corresponding weight $w_i$ (From $w0$ for the first layer up to $w11$ for the last layer.).

combines the two approaches. The BERT part of the model is not fine-tuned however the weights of the linear combination of embeddings from BERT layers are still learned.

### 4.5.2 JBERT

The joint JBERT model is the same as JPDRMM, but uses the BERT model for sentence retrieval (Section 4.4.2), instead of PDRMM, to obtain the initial sentence scores. The top layers of Figure 4.9 are then used, as in all joint models, to obtain the document score from the sentence scores and revise the sentence scores. Similarly to BJPDRMM, we also experimented with variations of JBERT, which do not fine-tune the parameters of BERT (JBERT-NF), use a linear combination (with trainable weights) of the '[CLS]' embeddings from all the BERT layers (JBERT-ADAPT), or both (JBERT-ADAPT-NF).

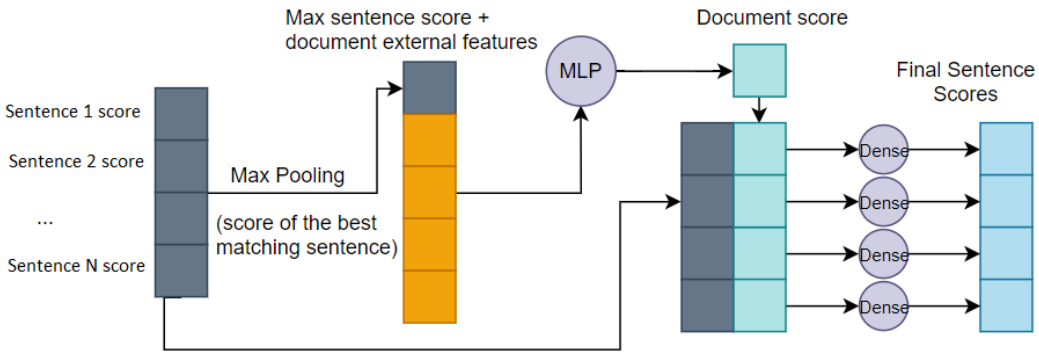


Fig. 4.9 Final layers of JPDRMM and JBERT. The input sentence scores are generated by PDRMM (Figure 4.2) or BERT (Figure 4.3) now applied to document *sentences*. The document's score is obtained from the score of its best sentence and external features and is also used to revise the sentence scores. Training jointly minimizes document and sentence loss.

## 4.6 Dense Retrieval

### 4.6.1 SEMantic Indexing for SEntence Retrieval (SEMISER)

Our JPDRMM-based models of the previous section rely on conventional information retrieval to obtain a set of $N$ possibly relevant documents from the document collection, and then invoke JPDRMM to re-rank the retrieved $N$ documents and their snippets. Instead, in this section, we have created a neural encoder to map each sentence of the document collection to a sentence embedding, and we index the sentences of the document collection by their sentence embeddings. We use a similar encoder to map each question to a question embedding, and approximate $k$-NN retrieval algorithms [21, 113] to retrieve the sentences of the document collection whose embeddings are closest to the question embedding; the retrieved sentences are ranked by increasing distance to the question embedding. When required to retrieve documents too, we simply report the documents that contained the retrieved sentences; the relevance score of each document is the minimum question-sentence distance over all the sentences of the document.[19] The encoder of the sentences and the encoder of the queries are jointly trained in a 'self-supervised' manner, detailed below, which does not require manually labeled gold relevant documents and snippets per training question. Our proposed method called SEMISER (SEMantic Indexing for SEntence Retrieval), is a new deep learning model for semantic indexing of sentences [200].

SEMISER takes a sentence and a question as input (Figure 4.10). Each word of the sentence and question is mapped to the corresponding word embedding. In BIOASQ 8, we

[19] We also maintain an index that maps sentences to their documents.

used the same biomedical WORD2VEC embeddings as in the JPDRMM methods.[20] Two stacked trigram convolutional layers (with tanh activations) are used to obtain a context-aware embedding for each word, and an attention layer (different for sentences and queries) then computes the sentence and question embeddings. The attention layer actually produces two sentence embeddings (vectors) and two question embeddings. The intuition is that the two vectors will capture different views of the sentence and question, respectively, similar in spirit to the multiple representations obtained when using multiple attention heads in Transformer-based models [37]. To force the two sentence (or question) embeddings to learn different views of the sentence (or question), we compute a cosine similarity loss between the two sentence (or question) embeddings during training. We also compute the maximum cosine similarity over all four pairs of sentence and question embeddings and require it to be 1 (or 0) when SEMISER is given a question and a relevant (or irrelevant) sentence, using binary cross-entropy loss.[21] In our experiments, we simply added the three losses. Although SEMISER can be trained in a supervised manner, by using pairs consisting of queries and relevant (or irrelevant) sentences as positive (or negative) training instances, BIOASQ provides relatively few training instances by today's standards (approx. 2.6*k* training queries, with approx. 1.24 relevant snippets per question on average). Instead, we opted for a 'self-supervised' approach, using an auxiliary training task for which very large numbers of training instances can be obtained without manual annotation.

For the auxiliary training task, we used sentences from 50k randomly selected PUBMED documents. The positive training instances were pairs consisting of one of the sentences and a (possibly multi-word) key term extracted from the sentence using SGRANK [40], an unsupervised key term extraction method. The negative training instances were pairs consisting of one of the sentences and a key term extracted from another randomly selected sentence. This process led to approx. 2.3 million training instances; we generated an equal number of positive and negative instances. In effect, the auxiliary task requires SEMISER to be able to generate sentence and query embeddings containing enough information to decide if a sentence contains a key term (treated as a query) or not.

The intuition is that in most cases relevant sentences contain key terms of the queries, hence being able to predict if a key term is included in a sentence is important. By forcing key terms (more generally queries) to be represented by low-dimensional embeddings, we also hope that similar queries will end up being close in the vector space and that similar sentences will also end up being close in a similar manner.

[20] The word embeddings are not updated during training.

[21] We replace all negative cosine similarity values with zeros, using a RELU activation.

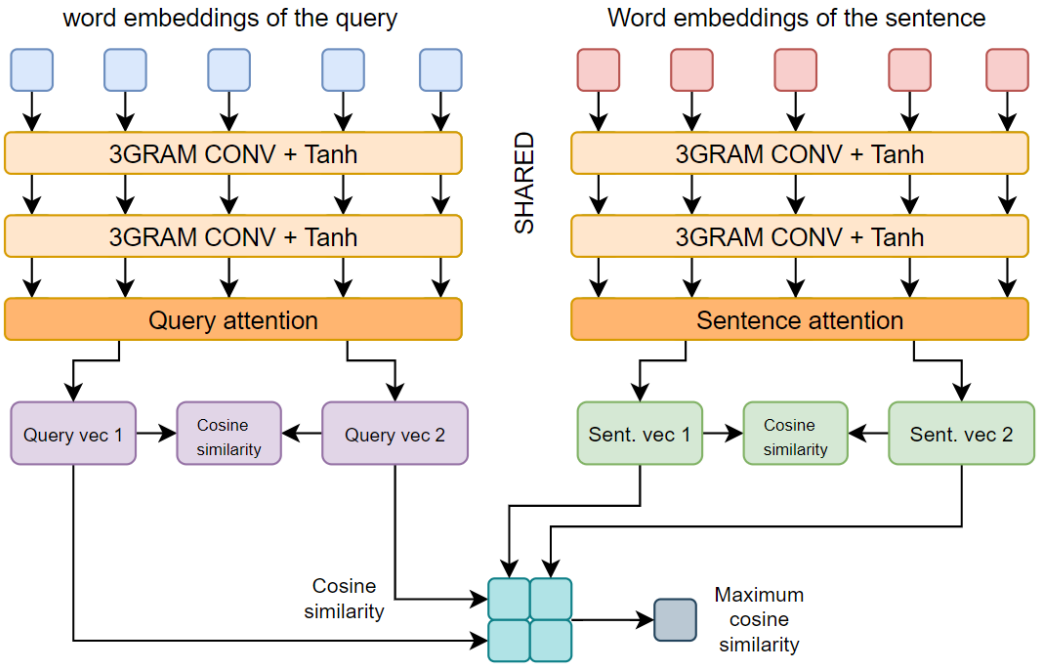


Fig. 4.10 Illustration of the SEMISER model. Two trigram convolutions are applied to the sentence embeddings (right) and an attention layer computes two vectors for the entire sequence. A first cosine similarity (green box) is computed between the two vectors of the sentence. During training, this similarity score is minimized so that the two vectors differ. The same process is applied to the query embeddings and two vectors are computed for the query. A cosine similarity is computed between the sentence vectors and the query vectors and the maximum similarity is used as an overall similarity score between the sentence and the query.

Having trained SEMISER, we use its right part (Figure 4.10) to obtain and index (off-line) sentence embeddings and the left part to convert queries (on the fly) to query embeddings. We use SEMISER to create two vector representations for each sentence in PUBMED. Both vectors of all sentences are stored in the same index since they are computed using the same convolutional layers. To retrieve sentences (and the documents that contain them), we query the index of sentence embeddings (using approximate $k$-NN matching) to obtain the sentences with the most similar sentence embeddings. For each retrieved sentence, we compute again the maximum similarity score over all four pairs of sentence-query embeddings.

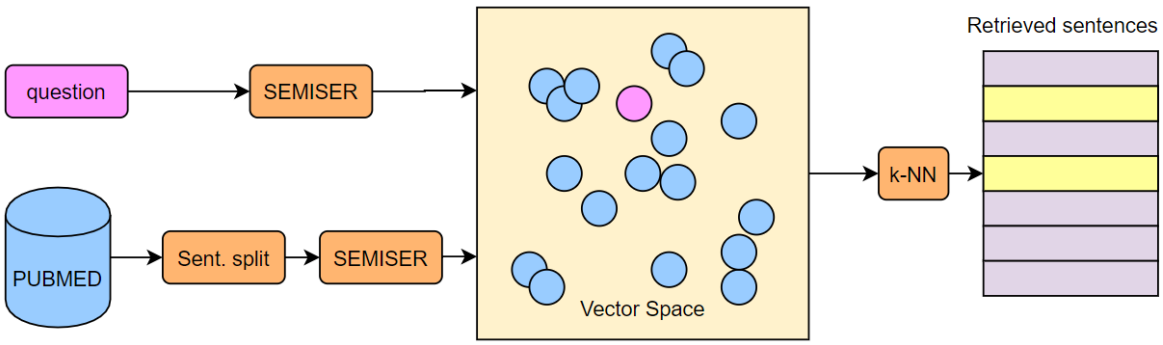


Fig. 4.11 Using SEMISER for snippet (sentence) retrieval. Each sentence of each PUBMED document is translated to two vector representations and all vectors are pre-computed and indexed. In order to retrieve sentences relevant to a question, two vector representations of the question are also computed and the sentences with the best cosine similarity in the vector space are selected as relevant. Documents inherit the scores of their best snippets.

## 4.7 Experiments

### 4.7.1 Experiments on BIOASQ-7

During this thesis we have participated in multiple BIOASQ contests. Using data from BIOASQ-7 (data provided by BIOASQ organizers during the seventh year of the competition) we trained several models for document retrieval and snippet extraction. In BIOASQ-7 we participated with TERM-PACRR, PDRMM and BERT for document retrieval and BCNN, ABCNN3, PDRMM, BCNN-PDRMM and ABCNN3-PDRMM for snippet extraction. We also report the results of a simple pipeline that used BM25 both for document retrieval and snippet extraction.

We submitted five different systems to BIOASQ 7 (Task 7b, Phase A), all of which consist of components described above.

**AUEB-NLP-1**: W2V-JPDRMM (Section 4.5.1) for both document retrieval and snippet extraction.

**AUEB-NLP-2**: BJPDRMM (Section 4.5.1) for both document retrieval and snippet extraction.

**AUEB-NLP-3**: Pipeline consisting of TERM-PACRR (Section 4.4.1) for document retrieval, followed by BCNN (Section 4.4.2) for snippet retrieval in batches 2 and 3[22], or PDRMM (Section 4.4.2) in batches 4 and 5.

[22] See section 3.2 for details on BIOASQ batches.

**AUEB-NLP-4**: Pipeline consisting of BERT for document retrieval, followed by BCNN (Section 4.4.2) for snippet retrieval in batches 2 and 3, or PDRMM (Section 4.4.2) in batches 4 and 5.

**AUEB-NLP-5**: Pipeline consisting of BERT high confidence for document retrieval, followed by BCNN (Section 4.4.2) for snippet retrieval in batches 2 and 3, or PDRMM (Section 4.4.2) in batches 4 and 5.

In all five systems, after obtaining the top 10 documents and top 10 snippets, we re-ranked the top 10 snippets by the scores of the documents they came from. The goal was to promote snippets coming from highly relevant documents. In the first two systems, which use JPDRMM versions, this final re-ranking of the snippets made almost no difference, since JPDRMM internally revises the scores of the snippets taking into account the scores of the documents they come from.

Table 4.1 reports the results of BIOASQ-7 tasks for document retrieval (Task b - phase A) and snippet extraction (Task b - phase B) for batches 2–5. We did not participate in batch 1. We observe that the BERT document ranker (used in AUEB-NLP-4) has the best document MAP scores in all batches.[23] The BERT high confidence document ranker (used in AUEB-NLP-5) has the second best document MAP overall, but with greatly improved F1.

Interestingly, the joint model (used in AUEB-NLP-1/2) outperformed comparable pipeline systems (AUEB-NLP-1 vs. AUEB-NLP-3, AUEB-NLP-2 vs. AUEB-NLP-4) by a wide margin in snippet MAP. It obtained very competitive results in snippet MAP even without using BERT embeddings (AUEB-NLP-1) and against pipelines that used BERT for document retrieval (AUEB-NLP-4) and additional re-ranking heuristics (AUEB-NLP-5). Recall, also, that in the joint model we selected the best training epoch by monitoring the *document* MAP on development data, whereas for the snippet retrieval components of the pipeline models (AUEB-NLP-3/4/5) *snippet* MAP was monitored; hence, the snippet MAP scores of the joint model might improve further by monitoring *snippet* MAP. We also note that the joint models use much fewer trainable parameters than the pipeline models (Table 4.2) and they outperform AUEB-NLP-3, which was one of the best systems of BIOASQ 6. It is also interesting that in both document and snippet retrieval, there is no clear difference between AUEB-NLP-1, which does not rely on BERT at all, and AUEB-NLP-2, which uses BERT to obtain word embeddings.

Particularly interesting is that the joint model (AUEB-NLP-1/2) outperforms the BERT based high-confidence model (AUEB-NLP-5) in snippet retrieval. Similarly to Brokos et al. [24], we observed that passing only high-confidence retrieved documents to the snippet ranking component in pipeline systems improved snippet retrieval greatly (compare the

[23] Document MAP is the official document retrieval measure of BIOASQ and also the measure we monitored on the development data to select the best training epoch.

snippet scores of AUEB-NLP-4 vs. AUEB-NLP-5), because it allowed the snippet retrieval component to operate only on documents that were likely to be relevant. However, JPDRMM did not require such heuristics. Instead, since it models the fact that good snippets come from good documents and vice-versa, it naturally selected snippets mostly from high-confidence documents. Thus the empirical results validate the hypothesis that joint modeling is beneficial. An open question is why the joint models do worse on document ranking compared to the pipeline models (AUEB-NLP-4/5). This is likely due to BERT (the document scorer of AUEB-NLP-4/5) being such a powerful model. A future line of investigation is to build joint models that integrate BERT to a larger extent, instead of just providing word embeddings to JPDRMM as in BJPDRMM.

### 4.7.2 Ablation studies on the joint document and snippet retrieval models

Ablation is a technique that involves removing or deactivating specific components of a deep neural network model to study the impact on model performance. This technique can be helpful for understanding the relative importance of different model components and for identifying potential areas for improvement. In this section, we report ablation studies and evaluate the impact of additional hand-crafted sentence and document features on the performance of the JPDRMM model.

We applied 10-fold cross-validation on the data of BIOASQ 7 and report the results in Table 4.3. All joint models discussed in Section 4.5.1 use the sum of the document and snippet loss ($L = L_{doc} + L_{snip}$). By contrast, in Table 4.3 we use a linear combination $L = L_{doc} + \lambda_{snip} L_{snip}$ and tune the hyper-parameter $\lambda_{snip} \in \{100, 10, 1, 0.1, 0.01\}$. We also try removing the extra document and/or sentence features to check their effect. This experiment was performed only with JPDRMM, which is one of our best joint models and computationally much cheaper than methods that employ BERT. Here we perform a 10-fold cross-validation on the union of the training and development subsets.

Table 4.3 shows that further performance gains (6.80 to 7.85 document MAP, 15.42 to 17.34 snippet MAP) are possible by tuning the weights of the two losses, comparing to the configuration we had used in Section 4.7.1, where both the sentence and document extra features were used and $\lambda_{snip}$ was 1. The best scores are obtained when using both the extra sentence and document features. However, the model performs reasonably well even when one of the two types of extra features is removed, with the exception of $\lambda_{snip} = 10$. The standard deviations of the MAP scores over the folds of the cross-validation indicate that the performance of the model is reasonably stable.

### 4.7.3 Experiments on BIOASQ-8

We submitted the following five systems to BIOASQ 8 (Task 8b, Phase A). In all cases, we used BM25 when scoring documents with ElasticSearch.

**AUEB-NLP-1**: W2V-JPDRMM (Section 4.5.1) for document and snippet retrieval, with BM25 (Section 4.4.2) for initial document retrieval.

**AUEB-NLP-2** (batches 2–5 only): Same as AUEB-NLP-1, but with GRAPH-JPDRMM (Section 4.5.1) instead of W2V-JPDRMM (Section 4.5.1).

**AUEB-NLP-3**: Same as AUEB-NLP-1, but we use SEMISER (Section 4.6.1) to re-score the sentences of the $n_d$ documents that W2V-JPDRMM (Section 4.5.1) retrieves. Each one of the $n_d$ documents is then re-ranked by the score of its best snippet.

**AUEB-NLP-4**: SEMISER (Section 4.6.1) for document and snippet retrieval (Figure 4.10) in batches 1–2. An ensemble of AUEB-NLP-1 and AUEB-NLP-2 in batches 3–5. In batches 3–4, the ensemble summed the scores of the two models (both when scoring documents and snippets); in batch 5, it used the maximum score of the two models.

**AUEB-NLP-5**: BM25 (Section 4.4.2) for document retrieval, then SEMISER for snippet retrieval.

The last three systems were intended to test the performance of SEMISER, when used on its own for both document and snippet retrieval (AUEB-NLP-4, batches 1–2), when pipelined after BM25 (AUEB-NLP-5), or when used as an additional re-scoring mechanism after W2V-JPDRMM (AUEB-NLP-3). Since SEMISER performed very poorly when used on its own (AUEB-NLP-4, batches 1–2), in the last three batches we used the slot of AUEB-NLP-4 to experiment with ensembles of our two best systems (AUEB-NLP-1 and AUEB-NLP-2).

Table 4.4 reports the official MAP scores of our systems for batches 1–5, along with the best score achieved by other participants in each batch. We also report system rankings, again based on MAP.

A first observation is that AUEB-NLP-1 and AUEB-NLP-2, which use W2V-JPDRMM and GRAPH-JPDRMM respectively, performed particularly well in snippet retrieval. In batches 1–4, they were the top two systems in snippet retrieval, largely outperforming all other systems in MAP, and they ranked 2nd and 3rd in batch 5, where their MAP was close to that of the best system.[24] The two systems also performed well in document retrieval, where they were ranked in the top 8 positions in all batches among more than 20 participants. These document and snippet retrieval results also indicate that JPDRMM works equally well with the original biomedical WORD2VEC embeddings (W2V-JPDRMM) and the word embeddings we obtained from the entity co-occurrence graph via the graph node embedding method

[24] We are surprised by the fact that the official MAP scores occasionally exceed 100%, which may be due to using a wrong normalization.

(GRAPH-JPDRMM see Section 4.5.1). We observe that in all batches AUEB-NLP-1 (i.e. the W2V-JPDRMM) is always located in the top 5 systems, excluding the document retrieval score of batch 5. AUEB-NLP-2 (i.e. the GRAPH-JPDRMM) competes with AUEB-NLP-1 in all batches and surpassed it in batch 4.

Another key observation is that SEMISER, which uses self-supervised neural encoders to index and retrieve sentences and indirectly documents (AUEB-NLP-4, batches 1–2 only), performed poorly, both in document and snippet retrieval. When SEMISER was used only to score the sentences of documents retrieved by BM25 (AUEB-NLP-5), its snippet MAP improved substantially (see batches 1–2), but remained well below the snippet MAP of the best systems. For document retrieval, AUEB-NLP-5 uses BM25, hence the corresponding document MAP results show the performance of conventional information retrieval. When SEMISER was used to re-score sentences and documents retrieved by BM25 and W2V-JPDRMM (AUEB-NLP-3), both document MAP and snippet MAP were lower than those of AUEB-NLP-5. Overall, we were unable to obtain benefits by including SEMISER in any of our systems. The simplistic ensembles (summing or taking the maximum score) of AUEB-NLP-1 and AUEB-NLP-2 that we experimented with (AUEB-NLP-4, batches 3–5) did not improve document MAP and, more surprisingly, led to much worse snippet MAP compared to the scores of the systems we combined.

### 4.7.4 Experiments on the Natural Questions Dataset

Even though there was no other large-scale IR dataset providing multiple gold documents and snippets per question, we needed to test our best models on a second dataset, other than BIOASQ. Therefore we modified the Natural Questions dataset [87] to a format closer to BIOASQ's. Each instance of Natural Questions consists of an HTML document of Wikipedia and a question. The answer to the question can always be found in the document as if a perfect retrieval engine were used. A short span of HTML source code is annotated by humans as a 'short answer' to the question. A longer span of HTML source code that includes the short answer is also annotated, as a 'long answer'. The long answer is most commonly a paragraph of the Wikipedia page. In the original dataset, more than 300,000 questions are provided along with their corresponding Wikipedia HTML documents, short answer and long answer spans. We modified Natural Questions to fit the BIOASQ setting. From every Wikipedia HTML document in the original dataset, we extracted the paragraphs and indexed each paragraph separately to an ElasticSearch index, which was then used as our retrieval engine. We discarded all the tables and figures of the HTML documents and any question that was answered by a paragraph containing a table. For every question, we apply a query to our retrieval engine and retrieve the first $N = 100$ paragraphs. We treat each paragraph as a

document, similarly to the BIOASQ setting. For each question, the gold (correct) documents are the paragraphs (at most two per question) that were included in the long answers of the original dataset. The gold snippets are the sentences (at most two per question) that overlap with the short answers of the original dataset. We discard questions for which the retrieval engine did not manage to retrieve any of the gold paragraphs in its top 100 paragraphs. We ended up with 110,589 questions and 2,684,631 indexed paragraphs. Due to a lack of computational resources, we only use 4,000 questions for training, 400 questions for development, and 400 questions for testing, but we make the entire modified Natural Questions dataset publicly available.

Table 4.5 reports results on the modified Natural Questions dataset. We experiment with the best pipeline and joint model of Table 4.1 that did not use BERT (and are computationally much cheaper), i.e., PDRMM+PDRMM and JPDRMM, comparing them to the more conventional BM25+BM25 baseline. Since there are at most two relevant documents and snippets per question in this dataset, we measure Mean Reciprocal Rank (MRR) [114], and Recall at top 1 and 2. Both PDRMM+PDRMM and JPDRMM clearly outperform the BM25+BM25 pipeline in both document and snippet retrieval. As in Table 4.1, the joint JPDRMM model outperforms the PDRMM+PDRMM pipeline in snippet retrieval, but the pipeline performs better in document retrieval. Again, this is unsurprising, since the joint models are geared towards snippet retrieval. We also note that JPDRMM uses half of the trainable parameters of PDRMM+PDRMM. No comparison to previous work that used the original Natural Questions is possible, since the original dataset provides a single document per question.

| DOCUMENT RETRIEVAL | | | | |
|---|---|---|---|---|
| **System** | **Rank** | **F-M.** | **MAP** | **GMAP** |
| Batch 2 | | | | |
| AUEB-NLP-1 | 9 | 17.84 | 7.41 | 0.66 |
| AUEB-NLP-2 | 10 | 18.23 | 7.41 | 0.62 |
| AUEB-NLP-3 | 5 | 19.05 | 7.71 | **0.75** |
| **AUEB-NLP-4** | 1 | 19.11 | **8.49** | 0.67 |
| AUEB-NLP-5 | 2 | **34.43** | 8.30 | 0.49 |
| Top Comp. | 3 | 18.77 | 7.91 | 0.48 |
| Batch 3 | | | | |
| AUEB-NLP-1 | 4 | 23.80 | 10.41 | 1.18 |
| AUEB-NLP-2 | 2 | 24.49 | 11.21 | 1.56 |
| AUEB-NLP-3 | 6 | 22.66 | 9.86 | 1.04 |
| **AUEB-NLP-4** | 1 | 24.71 | **11.99** | 1.51 |
| AUEB-NLP-5 | 3 | **40.34** | 11.02 | **1.64** |
| Top Comp. | 5 | 28.94 | 10.33 | 0.18 |
| Batch 4 | | | | |
| AUEB-NLP-1 | 4 | 20.56 | 9.51 | 1.01 |
| AUEB-NLP-2 | 3 | 20.51 | 9.68 | 0.83 |
| AUEB-NLP-3 | 5 | 19.42 | 9.09 | 0.83 |
| **AUEB-NLP-4** | 1 | 21.48 | **10.34** | 1.12 |
| AUEB-NLP-5 | 2 | **37.83** | 10.15 | **1.16** |
| Top Comp. | 6 | 18.53 | 8.35 | 0.51 |
| Batch 5 | | | | |
| AUEB-NLP-1 | 3 | 9.90 | 3.68 | 0.06 |
| AUEB-NLP-2 | 6 | 9.00 | 3.55 | 0.06 |
| AUEB-NLP-3 | 5 | 9.91 | 3.66 | 0.07 |
| **AUEB-NLP-4** | 1 | 11.20 | **4.25** | **0.10** |
| AUEB-NLP-5 | 2 | **20.12** | 3.99 | 0.08 |
| Top Comp. | 4 | 9.27 | 3.68 | 0.05 |

| SNIPPET RETRIEVAL | | | | |
|---|---|---|---|---|
| **System** | **Rank** | **F-M.** | **MAP** | **GMAP** |
| Batch 2 | | | | |
| **AUEB-NLP-1** | 1 | 18.55 | **14.38** | 0.19 |
| AUEB-NLP-2 | 3 | 17.64 | 12.90 | **0.28** |
| AUEB-NLP-3 | 6 | 11.11 | 6.25 | 0.13 |
| AUEB-NLP-4 | 5 | 10.96 | 6.43 | 0.14 |
| AUEB-NLP-5 | 2 | **19.01** | 13.62 | 0.23 |
| Top Comp. | 4 | 12.12 | 8.93 | 0.04 |
| Batch 3 | | | | |
| **AUEB-NLP-1** | 1 | 24.72 | **22.06** | 0.81 |
| AUEB-NLP-2 | 2 | **25.63** | 21.97 | **0.89** |
| AUEB-NLP-3 | 7 | 14.43 | 9.90 | 0.28 |
| AUEB-NLP-4 | 5 | 15.44 | 11.26 | 0.37 |
| AUEB-NLP-5 | 3 | 24.56 | 19.21 | 0.85 |
| Top Comp. | 4 | 16.17 | 14.04 | 0.09 |
| Batch 4 | | | | |
| AUEB-NLP-1 | 2 | 24.40 | 20.86 | 0.65 |
| **AUEB-NLP-2** | 1 | 23.65 | **21.14** | 0.75 |
| AUEB-NLP-3 | 7 | 17.79 | 11.49 | 0.53 |
| AUEB-NLP-4 | 9 | 17.91 | 11.16 | 0.56 |
| AUEB-NLP-5 | 3 | **24.67** | 18.21 | **0.98** |
| Top Comp. | 4 | 17.23 | 15.27 | 0.13 |
| Batch 5 | | | | |
| AUEB-NLP-1 | 3 | 8.04 | 5.81 | 0.02 |
| **AUEB-NLP-2** | 1 | 8.18 | **6.31** | **0.03** |
| AUEB-NLP-3 | 8 | 5.81 | 3.87 | 0.02 |
| AUEB-NLP-4 | 6 | 6.53 | 4.16 | 0.02 |
| AUEB-NLP-5 | 2 | **9.89** | 6.17 | **0.03** |
| Top Comp. | 4 | 6.56 | 4.99 | 0.01 |

Table 4.1 Performance on BIOASQ Task 7b, Phase A (batches 2–5) for document and snippet retrieval. Top Comp. is the top scoring submission from other teams.

| **Model** | **Number of Parameters** |
|---|---|
| AUEB-NLP-1 | 5,793 |
| AUEB-NLP-2 | 3,541,551 |
| AUEB-NLP-3 | 16,519 |
| AUEB-NLP-4/5 (with BCNN for snippets) | 109,499,902 |
| AUEB-NLP-4/5 (with PDRMM for snippets) | 109,489,455 |

Table 4.2 The number of trainable parameters for systems submitted in BIOASQ 7.

| Sent extra | doc extra | weight | AV. DOC MAP | AV. SENT MAP |
|---|---|---|---|---|
| yes | yes | 100 | 6.08 (0.17) | 14.50 (0.34) |
| yes | no | 100 | 1.34 (0.19) | 03.94 (0.49) |
| no | yes | 100 | 0.78 (0.26) | 01.73 (0.42) |
| yes | yes | 10 | 6.23 (0.14) | 14.73 (0.32) |
| yes | no | 10 | 1.20 (0.14) | 03.59 (0.45) |
| no | yes | 10 | 1.18 (0.23) | 02.19 (0.29) |
| *yes* | *yes* | *1* | *6.80 (0.07)* | *15.42 (0.23)* |
| yes | no | 1 | 1.35 (0.24) | 03.77 (0.73) |
| no | yes | 1 | 7.35 (0.16) | 14.58 (0.88) |
| yes | yes | 0.1 | 7.85 (0.08) | 17.28 (0.26) |
| yes | no | 0.1 | 6.77 (0.25) | 13.86 (1.10) |
| no | yes | 0.1 | 7.59 (0.12) | 15.77 (0.60) |
| yes | yes | 0.01 | 7.83 (0.07) | 17.34 (0.37) |
| yes | no | 0.01 | 6.61 (0.19) | 12.96 (0.29) |
| no | yes | 0.01 | 7.65 (0.10) | 14.24 (1.63) |

Table 4.3 JPDRMM cross-validation results on BIOASQ 7 data for tuned weights of the two losses, with and without the extra sentence and document features. The MAP scores are averaged over the 10 folds. We also report standard deviations ($\pm$). (Results corresponding to the configuration of Section 4.7.1 are shown in italics)

| **DOCUMENT RETRIEVAL** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **System** | **Rank** | **MAP** | **Rank** | **MAP** | **Rank** | **MAP** | **Rank** | **MAP** | **Rank** | **MAP** |
| Batch | Batch 1 | | Batch 2 | | Batch 3 | | Batch 4 | | Batch 5 | |
| AUEB-NLP-1 | 3 | 33.59 | 2 | 31.81 | 3 | 44.41 | 5 | 40.09 | 8 | 45.97 |
| AUEB-NLP-2 | n/a | n/a | 6 | 31.03 | **1** | **45.1** | **1** | **41.63** | 6 | 46.57 |
| AUEB-NLP-3 | 14 | 24.37 | 14 | 27.02 | 24 | 32.56 | 25 | 28.29 | 23 | 33.89 |
| AUEB-NLP-4 | 21 | 4.93 | 26 | 7.12 | 2 | 45.02 | 2 | 41.47 | 9 | 45.96 |
| AUEB-NLP-5 | 13 | 28.62 | 9 | 28.43 | 15 | 38.31 | 19 | 34.77 | 18 | 40.93 |
| Top Comp. | **1** | **33.98** | **1** | **33.04** | 4 | 43.69 | 3 | 41.21 | **1** | **48.42** |
| **SNIPPET RETRIEVAL** | | | | | | | | | | |
| **System** | **Rank** | **MAP** | **Rank** | **MAP** | **Rank** | **MAP** | **Rank** | **MAP** | **Rank** | **MAP** |
| Batch | Batch 1 | | Batch 2 | | Batch 3 | | Batch 4 | | Batch 5 | |
| AUEB-NLP-1 | **1** | **85.75** | **1** | **68.21** | 2 | 96.32 | **1** | **102.44** | 2 | 108.31 |
| AUEB-NLP-2 | n/a | n/a | 2 | 65.49 | **1** | **100.39** | 2 | 99.2 | 3 | 106.39 |
| AUEB-NLP-3 | 9 | 21.71 | 15 | 15.56 | 13 | 23.85 | 13 | 17.54 | 13 | 26.37 |
| AUEB-NLP-4 | 15 | 2.73 | 17 | 3.28 | 9 | 35.94 | 11 | 34.24 | 12 | 33.56 |
| AUEB-NLP-5 | 4 | 36.36 | 10 | 22.17 | 10 | 32.17 | 8 | 35.31 | 11 | 37.34 |
| Top Comp. | 2 | 54.49 | 3 | 33.74 | 3 | 65.58 | 3 | 71.63 | **1** | **112.67** |

Table 4.4 Performance on BIOASQ Task 8b, Phase A for document and snippet retrieval. Top Comp. is the top scoring submission of other teams. AUEB-NLP-2 (W2V-JPDRMM) did not participate in batch 1. AUEB-NLP-4 was different in batches 1–2 (SEMISER on its own) and batches 3–5 (W2V-JPDRMM and GRAPH-JPDRMM ensembles).

| | | **Document Retrieval** | | | **Snippet Retrieval** | | |
|---|---|---|---|---|---|---|---|
| **Method** | **Parameters** | **MRR** | **Recall@1** | **Recall@2** | **MRR** | **Recall@1** | **Recall@2** |
| BM25+BM25 | 4 | 30.18 | 16.50 | 29.75 | 8.19 | 3.75 | 7.13 |
| PDRMM+PDRMM | 11.4k | **40.33** | **28.25** | **38.50** | 22.86 | 13.75 | 22.75 |
| JPDRMM | 5.8k | 36.50 | 24.50 | 36.00 | **26.92** | **19.00** | **25.25** |

Table 4.5 MRR (%) and recall at top 1 and 2 (%) on the modified **Natural Questions** dataset.

## 4.8 Deployment of Models in Vivo

In collaboration with Brown University and the Athena Research and Innovation Center, we have deployed one of our deep learning models and developed a literature identification system for systematic reviews. A systematic review is a summary of the medical literature, seeking to identify and synthesize all relevant information to formulate the best approach to diagnosis or treatment. Thousands of scientific articles are screened for each systematic review which makes it a cumbersome task for human experts. These articles are mainly retrieved using handcrafted queries with combinations of terminology using 'AND' and 'OR' rules.

In the domain of systematic reviews, a key question (Figure 4.12) refers to a clear and specific question that guides the review process. The key question serves as the foundation for defining the inclusion and exclusion criteria for the studies, as well as for analyzing and synthesizing the results. The key question should be developed based on the review's purpose and should reflect the underlying research problem. In systematic reviews, the key question is usually formulated using the PICO (population, intervention, comparison, outcome) framework, which helps to clarify the specific elements of the review question. The key question is essential for ensuring the validity, transparency, and replicability of the systematic review. As seen in Figure 4.12 these questions are not trivial and differ in length and content compared to questions used in BIOASQ.

**Table 1.** Key Question to Natural-Language Question Translation

| Key question | Natural-language question |
|---|---|
| **KQ 1:** What are the (comparative) benefits and harms of interventions to prevent attacks of primary headache in women who are pregnant (or attempting to become pregnant), postpartum, or breastfeeding? | What are the comparative benefits and harms of interventions *to prevent* attacks of primary headache in women who have a history of primary headache and are pregnant or attempting to become pregnant, postpartum, or breastfeeding? |
| **KQ 1a.** Do the (comparative) benefits and harms vary by phase (i.e., preconception, first trimester of pregnancy, second trimester of pregnancy, and third trimester of pregnancy, postpartum, breastfeeding)? | Do the comparative benefits and harms vary by phase (preconception, first trimester of pregnancy, second trimester of pregnancy, third trimester of pregnancy, postpartum, breastfeeding)? |
| **KQ 1b.** Do the (comparative) benefits and harms vary by type of primary headache (i.e., migraine, tension headache, cluster headache, and other trigeminal autonomic cephalgias)? | Do the comparative benefits and harms vary by type of primary headache (migraine, tension headache, cluster headache, and other trigeminal autonomic cephalgias)? |
| **KQ 2:** What are the (comparative) benefits and harms of interventions to treat acute attacks of primary headache in women who are pregnant (or attempting to become pregnant), postpartum, or breastfeeding? | What are the comparative benefits and harms of interventions to treat acute attacks of primary headache in women who are pregnant, attempting to become pregnant, postpartum, or breastfeeding? |
| **KQ 2a.** Do the (comparative) benefits and harms vary by phase (i.e., preconception, first trimester of pregnancy, second trimester of pregnancy, and third trimester of pregnancy, postpartum, breastfeeding)? | Do the comparative benefits and harms vary by phase (preconception, first trimester of pregnancy, second trimester of pregnancy, third trimester of pregnancy, postpartum, breastfeeding)? |
| **KQ 2b.** Do the (comparative) benefits and harms vary by type of primary headache (i.e., migraine, tension headache, cluster headache, and other trigeminal autonomic cephalgias)? | Do the comparative benefits and harms vary by type of primary headache (migraine, tension headache, cluster headache, and other trigeminal autonomic cephalgias)? |

*Abbreviations*: KQ, key question.

Fig. 4.12 Key questions posed by human experts working on abstract screening for systematic reviews. The questions are used unchanged as input to our retrieval model.

We developed a system called Pythia [3] which enables interactive abstract screening throughout the entire PUBMED database. Its functionality is based on an already-developed tool used by experts [189] for abstract screening but we propose an improved version where deep learning relevance ranking is integrated. The Key Questions are expressed in plain text, are then processed by Pythia, and a batch of 100 PUBMED abstracts are retrieved for screening. The user can then screen all relevant and irrelevant documents in the batch. Pythia takes into account human annotations of all relevant and irrelevant documents in the batch and presents the next batch of documents to the user. This process is repeated until users have collected all relevant to the systematic review articles.

Given a key question the system searches for candidate publications in the ElasticSearch database and retrieves the top $1,000$ documents based on BM25 (see section 2.1) similarity score. The documents are then re-ranked using scores computed by the JPDRMM model described in section 4.5. Since multiple key questions are submitted to the system we retrieve and rank 1000 abstracts for each key question and form a larger ranked list from the union of the retrieved abstracts of all key questions. If an abstract is retrieved by more than one key question the final score of the abstract is the maximum score computed. Finally, the top 100 abstracts are presented as the first batch of results to the users.

The abstracts are screened by a human, who annotates each one as relevant or irrelevant. Once this is done, Pythia extracts a set of positive key phrases from the publications annotated as relevant and a set of negative key phrases from the publications annotated as irrelevant, using SGRANK [40]. Any abstract examined by the users is removed from the pool of retrieved abstracts. The score of the remaining abstracts is penalized if any negative key phrase is present in the abstract or increased for any abstracts containing a positive key phrase, thus incorporating the feedback provided by users. Finally, the remaining abstracts are ranked based on their score and the next batch of 100 abstracts are presented to the user. This process is repeated until the user has retrieved a sufficient number of relevant documents.

Overall as human experts reported that in four reviews, the number of abstracts reviewed per relevant abstract number needed to read (NNR) was lower than in the manually screened project. On the other hand, our system did not perform well in two reviews, probably because the key questions do not resemble the data used to train the retrieval model (BIOASQ data). Finally, we observed that more relevant citations are retrieved in early batches, but retrieval was generally unaffected by other aspects, such as study design, study size, and specific key questions.

## 4.9 A demonstrator for searching in COVID-19 literature

The outbreak of the COVID-19 pandemic has led to many scientific publications. As more research was conducted in order to understand the new virus and identify new therapies thousands of new publications are published (Figure 4.13).

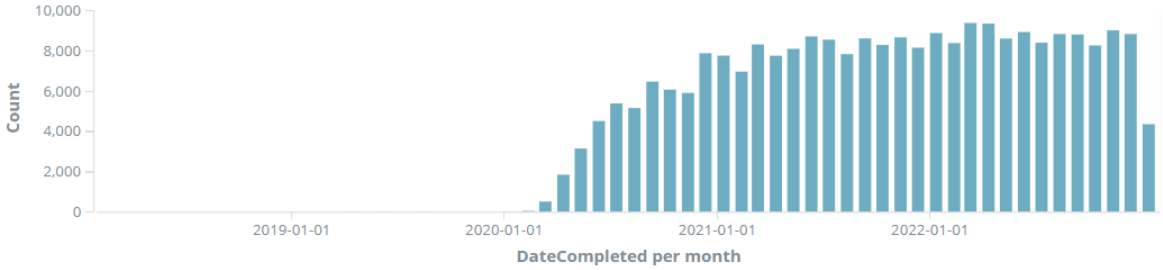


Fig. 4.13 COVID-19 related papers per month for the last 5 years.

We developed a prototype search engine for biomedical literature that can be used to aid researchers and medical professionals find the latest and most relevant information on COVID-19. The prototype retrieval engine for biomedical literature plays a crucial role in helping researchers and medical professionals stay informed about the latest advancements in COVID-19 research. This is achieved by offering quick and efficient access to an extensive collection of scientific articles on the subject. This not only saves valuable time, but also helps to keep professionals informed about the latest discoveries, treatments, and potential solutions to COVID-19.

In addition to its role in supporting COVID-19 research, the engine also helps to combat the spread of misinformation and fake news[71, 107, 159]. With so much information and misinformation being shared on the internet, it can be difficult for professionals and the general public to differentiate between credible sources and those that are not. The engine's curated collection of scientific articles, however, provides a trusted source of information on COVID-19, allowing professionals to make informed decisions and take action based on the most up-to-date research available.

We extracted approximately $1.8M$ sections from the full body of $190K$ biomedical articles openly available as the CORD-19 dataset [191]. To make our search engine robust we use our JPDRMM model described in section 4.5.1 to retrieve documents and relevant snippets.

The search engine is available through a web page [25]. On the landing page of our search engine, the user can express in plain text a question (Figure 4.14, first green box). Optionally the user can narrow down the results by filtering sentences that are only found in a specific

[25] Visit `http://cslab241.cs.aueb.gr:5000/`

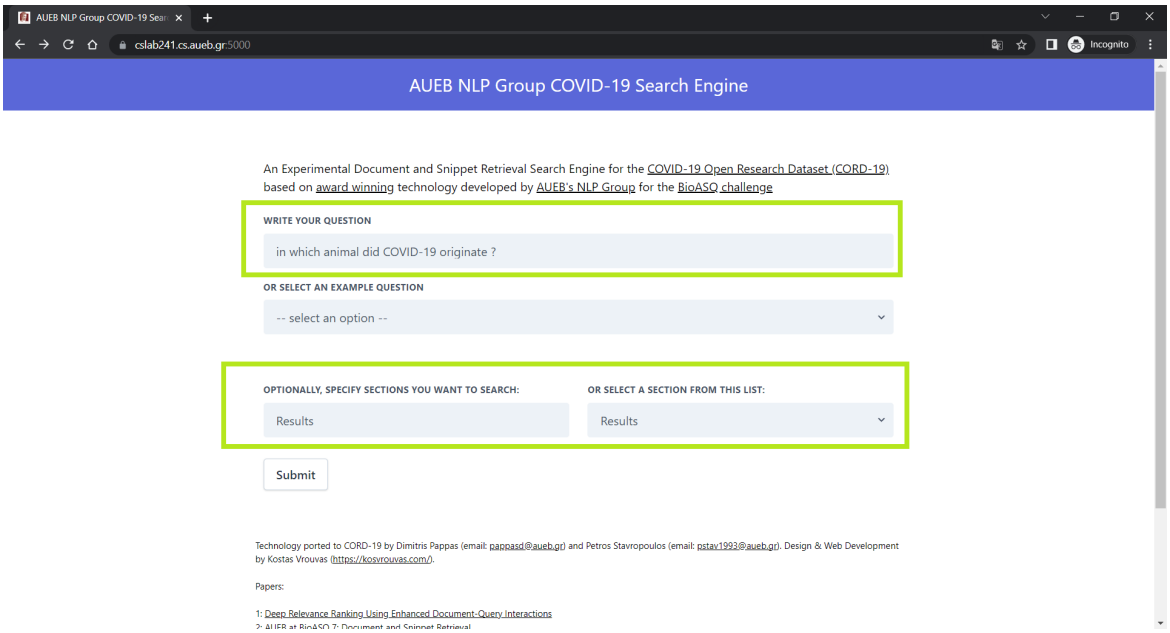


Fig. 4.14 Demonstration of search engine landing page.

section of the full body of the paper such as 'Introduction', 'Methods', 'Results' (second green box in Figure 4.14).

The search engine ranks the available sections in our corpus using JPDRMM and retrieves the top 20 documents (Figure 4.15). The graphical user interface assigns darker green colors to documents with higher relevance scores. The user views the titles of the papers that may include the answer to the question and can select one or many of them by clicking on the green box. Upon clicking a green box, the sentences of the relevant section appear (Figure 4.16). If one of the top relevant sentences is present in the document, the sentence is highlighted with yellow color.

Lastly, by clicking on a link that is created using the Digital Object Identifier (DOI), the user has the ability to be directed to the webpage of the original article on the publisher's website. Up to this date, more than $3.6K$ searches have been submitted to our COVID-19 search engine.

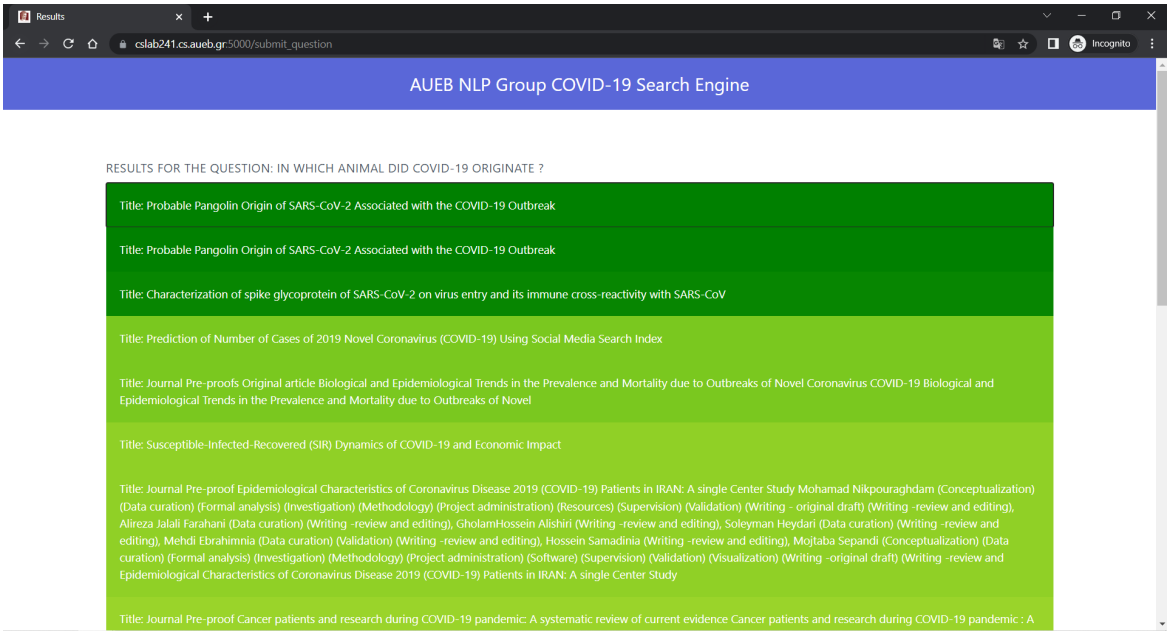


Fig. 4.15 Demonstration of search engine result page.

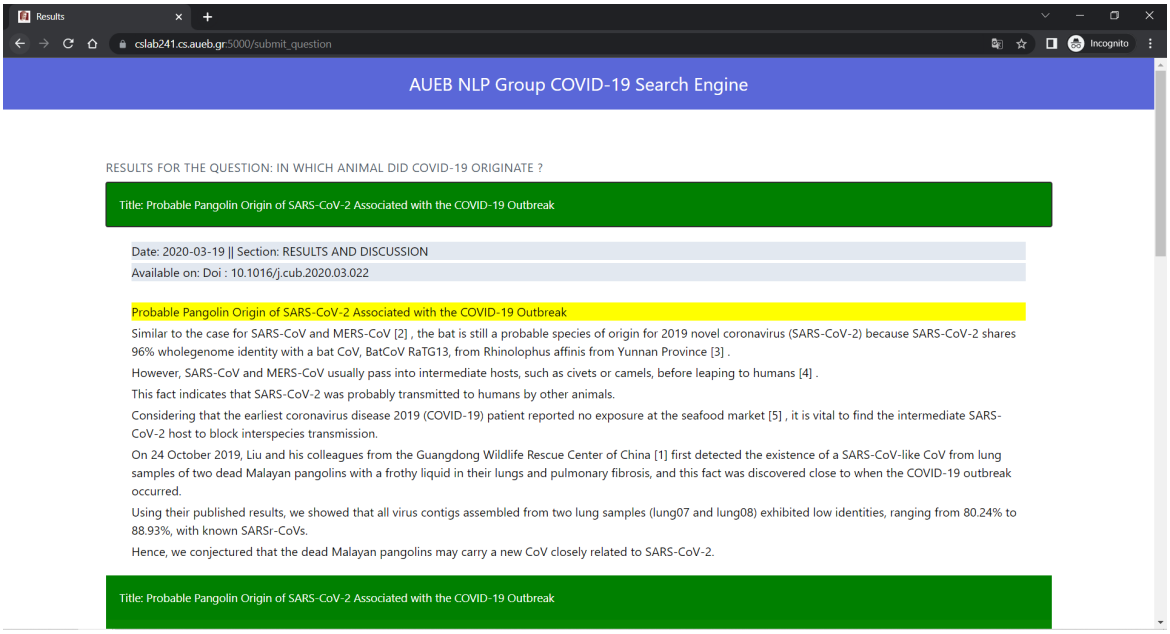


Fig. 4.16 Demonstration of search engine document selection.

## 4.10 Conclusions and Future Work

In this chapter, we presented our work on a new joint model for document and snippet retrieval in the BIOASQ competition. Our work makes several important contributions to the field of QA for large document collections. We proposed deep learning architectures to jointly rank documents and snippets with respect to a question. Our experiments showed that the joint models greatly outperform the traditional pipelines in snippet retrieval while remaining competitive in document retrieval. Additionally, we showed that our joint model that does not use BERT is competitive with BERT-based models and outperforms the best BIOASQ 7, and 8 systems.

We tested our retrieval models in the real-life scenario of systematic reviews. Our models helped experts retrieve relevant articles for a systematic review faster than traditional retrieval in four systematic reviews. Furthermore, we created a search engine for Covid-19 which can help biomedical experts submit a question and retrieve relevant documents. More than $3.6K$ requests have been processed up to this date.

We also provided a modified version of the Natural Questions dataset that is suitable for document and snippet retrieval. Our JPDRMM model performed well in both document and snippet retrieval on this dataset, improving the traditional BM25 ranking. Our retrieval models can be extended with code from chapter 6 of this thesis to extract factoid answers from the retrieved documents.

A major component of PDRMM-based models is the similarity scoring of documents and questions. Instead of replacing WORD2VEC in JPDRMM with BERT embeddings to create BJPDRMM or graph embeddings to create GRAPH-JPDRMM we can use all of the vector representations and let the network to decide which similarities are important. We have created a multi-view JPDRMM model (MV-JPDRMM) which uses seven similarity matrices instead of three. As vector representations, we use the one-hot encoding, the WORD2VEC embeddings, the graph embeddings, the BERT embeddings as well as contextual representation produced by three convolutional layers consuming the embeddings of WORD2VEC, graph, and BERT respectively. This approach resembles models like ABCNN3-PDRMM and BCNN-PDRMM used for snippet extraction in BIOASQ-7, but in MV-JPDRMM we combine all contextual similarities and ask the model to learn how to combine this information. This model has not yet been used in any competition, but we hope to do so in future work.

We hope to extend our joint models and datasets to also identify exact answer spans within snippets (Chapter 5), similar to the answer spans of SQUAD [154] and SQUAD v.2 [153]. This would lead to a multi-granular retrieval task, where systems would have to retrieve relevant documents, relevant snippets, and exact answer spans from the relevant snippets. BIOASQ already includes this multi-granular task, but exact answers are provided

only for factoid questions and they are rewritten by humans (as in MS-MARCO), rather than being spans of the relevant snippets.

# Chapter 5

# Biomedical Machine Reading Comprehension and New Artificial Datasets

## 5.1 Introduction

The human-annotated corpora have a clear and significant advantage in terms of the quality of the data. The data is meticulously curated and verified, ensuring that the annotations are accurate and consistent. This leads to a higher level of reliability and trust in the data, which is essential for various applications such as machine learning and artificial intelligence.

However, the creation of human-annotated data is a major challenge. It requires a significant amount of resources, both in terms of time and financial investment. Collecting and annotating large datasets can take months or even years, and the cost of paying human annotators can quickly add up. This results in human-annotated datasets that may not have the desired size or scale, especially for the training of deep learning models that require vast amounts of data to function optimally.

Moreover, the lack of scalability of the human annotation process creates a bottleneck for data collection, as it can be challenging to increase the size of the annotated dataset as needed. This is in contrast to automatically generated data, which can be produced at a much faster pace and at a lower cost. Therefore, while the quality of human-annotated data is unrivaled, the cost and time constraints associated with its creation may limit its practical use in some cases.

In order to enable researchers to test and implement new deep learning methods, we created two new large datasets (namely BioRead and BIOMRC) for biomedical cloze-style

Question Answering without the need for human annotations. A cloze-style question is a type of fill-in-the-blank question where certain words or phrases are removed from a sentence or a passage, and the reader is asked to fill in the missing words based on the context. The goal of a cloze-style question is to assess the reader's understanding of the text and their ability to use context clues to determine the missing information. An example of a cloze-style question from the CBTEST dataset can be seen in table 3.3 (see chapter 3).

In our research, we thoroughly evaluated multiple existing Machine Reading Comprehension (MRC) models on our newly created datasets and compare the results against strong baselines. The evaluation was conducted to assess the performance of these models in understanding and answering questions about the text. In a small-scale experiment, our models managed to outperform human performance in biomedical MRC when humans lack expertise in the biomedical domain but fail to surpass biomedical experts in the same task.

## 5.2 Related Work

Several biomedical MRC datasets exist [18], but have orders of magnitude fewer questions than the datasets we created and present in this chapter or are not suitable for a cloze-style MRC task [138, 19, 223]. The closest dataset to the datasets we created in this chapter is CLICR [180] (see also chapter 3), a biomedical MRC dataset with cloze-type questions created using full-text articles from case reports. The creators of the Clicr dataset used CLAMP [176] to detect biomedical entities and link them to concepts of the UMLS Metathesaurus [103]. Cloze-style questions (100$k$ passage-question instances) were created from the 'learning points' (summaries of important information) of the reports, by replacing biomedical entities with placeholders.

The QA dataset of BIOASQ [184] contains questions written by biomedical experts. / The gold answers comprise multiple relevant documents per question, relevant snippets from the documents, exact answers in the form of entities, as well as reference summaries, written by the experts. Creating data of this kind, however, requires significant expertise and time. In the first eight years of BIOASQ, only $3,243$ questions and gold answers had been created. So in chapter 6 we also explore if our new larger automatically generated datasets could be used to pre-train models, which could then be fine-tuned for human-generated QA or MRC datasets.

Outside the biomedical domain, several cloze-style open-domain MRC datasets have been created automatically such as CBTEST [62], CNN DailyMail [61], SearchQA [50], or BookTest [11], but have been criticized of containing questions that can be answered by simple heuristics like our basic baselines [30].

For example, Chen et al. [30] train a model that takes into account 8 features for each candidate to answer cloze-style questions of the DailyMail dataset. These features include the occurrence of an entity in the passage and the cloze-style question, the frequency of the entity in the passage, the first position of occurrence, the exact match between the text surrounding the entity and the surrounding text of the placeholder, word distance between the entity and non-stop question words, as well as dependency parse match between the placeholder and the entity. They show that even though the model is simple and uses these heuristic features it manages to compete or even surpass previously proposed neural network models for cloze-style QA.

There are also several large open-domain MRC datasets annotated by humans such as [87], Squad V1 [154], and V2 [153], NewsQA [183], MSMARCO [132], or the RACE dataset [88]. To our knowledge the biggest human-annotated corpus is Google's Natural Questions dataset [87] (see also chapter 4), with approximately $300K$ human-annotated examples. Datasets of this kind require extensive annotation effort, which for open-domain datasets is usually crowd-sourced. Crowd-sourcing, however, is much more difficult for biomedical datasets, because of the required expertise of the annotators.

## 5.3 New Datasets for Biomedical Machine Reading Comprehension

### 5.3.1 BioRead

We have created a new publicly available cloze-style biomedical machine reading comprehension (MRC) dataset with approximately 16.4 million passage-question instances called BioRead [141]. BioRead was constructed in the same way as the widely used Children's Book Test [62] and its extension BookTest [11] (see chapter 3), but using biomedical journal articles and employing MetaMap [10] to identify UMLS concepts. BioRead is one of the largest MRC datasets, and one of the largest in the biomedical domain.

PUBMED is a free search engine accessing primarily the MEDLINE database of references and abstracts on life sciences and biomedical topics. To create BioRead we randomly selected approximately $90.6k$ from the approximately $3.4M$ articles of the Open Access Subset of PUBMED Central and applied MetaMap [10] on them. MetaMap detects biomedical entities in a text, assigns a preferred name to the detected entity,[1] and links them to established biomedical ontologies by reference ids. We replaced each concept that MetaMap recognized by its preferred name. Using a shifting window of 21 sentences in each document, we

[1] E.g. MetaMap uses 'Carcinoma of lung' as the preferred name for 'lung cancer'.

removed an entity from the 21st sentence creating a pseudo-question and kept the rest 20 sentences as context. The entity was only removed when it could also be found in the context. We train deep learning models to select the entity found in the context which fills the placeholder in the pseudo-question. An instance example can be seen in Table 5.1.

| | |
|---|---|
| **Context:** *salsolinol* (100mg/kg i.p.) or *l-dopa* (100mg/kg i.p.) was acutely administered (100mg/kg i.p.). in the combined treatment group, *l-dopa* (100mg/kg i.p.) was administered once 15min after *salsolinol* administration. the *rats* were decapitated 2h after *injection*. the *concentration* of *dopamine* and its metabolites were measured using *hplc* . the results are expressed as the means sem (n=710 animals per group). the data were [...] *l-dopa* (f[1,27]=26.9, p<0.01) on the level of 3-mt (table1). however, neither *treatment* with *salsolinol* (f[1,27]=0.09, n.s.) nor the interaction between *salsolinol* and *l-dopa* (f[1,27]=0.03, n.s.) was significant (table1). | **Context:** *@entity10* (100mg/kg i.p.) or *@entity5* (100mg/kg i.p.) was acutely administered (100mg/kg i.p.). in the combined treatment group, *@entity5* (100mg/kg i.p.) was administered once 15min after *@entity10* administration. the *@entity6* were decapitated 2h after *@entity0* the *@entity3* of *@entity7* and its metabolites were measured using *@entity9* the results are expressed as the means sem (n=710 animals per group). the data were [...] *@entity5* (f[1,27]=26.9, p<0.01) on the level of 3-mt (table1). however, neither *@entity2* with *@entity10* (f[1,27]=0.09, n.s.) nor the interaction between *@entity10* and *@entity5* (f[1,27]=0.03, n.s.) was significant (table1). |
| **Question:** the duncans post hoc test showed that *l-dopa* induced an increase in the *concentration* of 3-mt (by approximately 300%, p<0.01) but that *salsolinol* did not influence this effect of *l-dopa* (table1). | **Question:** the duncans post hoc test showed that *@placeholder* induced an increase in the *@entity3* of 3-mt (by approximately 300%, p<0.01) but that *@entity10* did not influence this effect of *@entity5* (table1). |
| **Candidates:** injection, control group, treatment, concentration, substantia nigra, l-dopa, rats, dopamine, dopac, hplc, salsolinol, analysis | **Candidates:** @entity0, @entity1, @entity2, @entity3, @entity4, @entity5, @entity6, @entity7, @entity8, @entity9, @entity10, @entity11 |
| **Answer:** l-dopa | **Answer:** @entity5 |

Table 5.1 An example instance of BioRead, before (left) and after (right) replacing recognized UMLS concepts by pseudo-tokens. Red words and phrases are wrong candidate answers. The correct answer is shown in green and underlined.

We created two different versions of the dataset using two settings. In the first setting (setting A) we use a random identifier to replace an entity in an instance,(an example instance is shown in Table 5.1). If the same entity could be found in other instances of the dataset most probably this entity would have a different identifier. This forces the models to rely only on the context of an entity to select the best candidate to fill in the placeholder. In the second setting (setting B) we use a random identifier to replace an entity in the *entire* dataset.

| | BioRead | | | | BioReadLite | | | |
|---|---|---|---|---|---|---|---|---|
| | **Training** | **Development** | **Test** | **Total** | **Training** | **Development** | **Test** | **Total** |
| **Instances** | ∼15,1M | ∼600,7k | ∼652,9k | ∼16.4M | 800k | 50k | 50k | 900k |
| **Avg candidates** | 25.9 | 27.3 | 26.3 | 26.0 | 18.89 | 20.8 | 19.4 | 19.0 |
| **Max candidates** | 40 | 40 | 40 | 40 | 30 | 30 | 30 | 30 |
| **Min candidates** | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **Avg context len.** | 456.9 | 464.5 | 455.9 | 457.1 | 317.2 | 320.8 | 298.9 | 316.4 |
| **Max context len.** | 999 | 999 | 999 | 999 | 400 | 400 | 400 | 400 |
| **Min context len.** | 26 | 56 | 48 | 26 | 30 | 30 | 30 | 30 |
| **Avg question len.** | 33.4 | 35.5 | 34.8 | 33.5 | 16.8 | 16.8 | 16.8 | 16.8 |
| **Max question len.** | 300 | 300 | 300 | 300 | 25 | 25 | 25 | 25 |
| **Min question len.** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |

Table 5.2 Statistics of BioRead and BioReadLite. All lengths are measured in tokens using a whitespace tokenizer.

This allows our models to train their parameters based on the context and the entities as well. In that case, though, the models may not perform well if an entity that can be found in the test data has not been seen in the train data of the models. We also created BioReadLite, a subset of BioRead containing 900k instances, being thus suitable for research groups with fewer resources. Statistics for the two datasets are presented in Table 5.2. Both BioRead and BioReadLite are publicly available.[2]

### 5.3.2 BIOMRC

BIOMRC is a dataset very similar to BioRead in both nature and the creation process. bcContrary to BioRead which was created using the full body of biomedical articles, we used titles and abstracts from biomedical articles to create BIOMRC. BIOMRC alleviates several issues detected in BioRead. Many instances of BioRead contain passages or questions crossing article sections, originating from the references sections of articles, or including captions and footnotes due to parsing failure of the original articles (Table 5.3). Another source of noise is misclassified entities from MetaMap, which often misses or mistakenly identifies biomedical entities[3].

In BIOMRC to avoid crossing sections we use abstracts and titles of biomedical articles as passages and questions, respectively, which are clearly marked up in PUBMED data, instead of using the full text of the articles. Titles are likely to be related to their abstracts, which reduces the noise-to-signal ratio significantly and makes it less likely to generate irrelevant questions for a passage. We replace a biomedical entity in each title with a placeholder, and we require systems to guess the hidden entity by considering the entities of the abstract as

[2] `https://archive.org/details/bioread_dataset.tar`

[3] E.g., it often annotates 'to' as the country 'Togo'.

| |
|---|
| **'question' originating from caption**: |
| "figure 4 htert @entity6 and @entity4 XXXX cell invasion." |
| **'question' originating from reference**: |
| "2004 , 17 , 250 257 .14967013 c samuni y. ; samuni u. ; goldstein s. the use of cyclic XXXX as hno scavengers ." |
| **'passage' containing captions**: |
| "figure 2: distal UNK showing high insertion of rectum into common channel. figure 3: illustration of the cloacal malformation. figure 4: @entity5 showing UNK" |

Table 5.3 Examples of noisy BioRead data. XXXX is the placeholder, and UNK is the 'unknown' token. The first example is a question originating from a figure caption. The second example is a question originating from a reference section. The third example comprises text extracted from a reference section. These instances are unreliable as there is no assurance that they are contextually relevant to the surrounding text used to create the machine reading comprehension examples.

| | BIOMRC LARGE | | | | BIOMRC LITE | | | | BIOMRC TINY | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Training** | **Development** | **Test** | **Total** | **Training** | **Development** | **Test** | **Total** | **Setting A** | **Setting B** | **Total** |
| **Instances** | 700,000 | 50,000 | 62,707 | 812,707 | 87,500 | 6,250 | 6,250 | 100,000 | 30 | 30 | 60 |
| **Avg candidates** | 6.73 | 6.68 | 6.68 | 6.72 | 6.72 | 6.68 | 6.65 | 6.71 | 6.60 | 6.57 | 6.58 |
| **Max candidates** | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 13 | 11 | 13 |
| **Min candidates** | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 |
| **Avg abstract len.** | 253.79 | 257.41 | 253.70 | 254.01 | 253.78 | 257.32 | 255.56 | 254.11 | 248.13 | 264.37 | 256.25 |
| **Max abstract len.** | 543 | 516 | 511 | 543 | 519 | 500 | 510 | 519 | 371 | 386 | 386 |
| **Min abstract len.** | 57 | 89 | 77 | 57 | 60 | 109 | 103 | 60 | 147 | 154 | 147 |
| **Avg title len.** | 13.93 | 14.28 | 13.99 | 13.96 | 13.89 | 14.22 | 14.09 | 13.92 | 14.17 | 14.70 | 14.43 |
| **Max title len.** | 51 | 46 | 43 | 51 | 49 | 40 | 42 | 49 | 21 | 35 | 35 |
| **Min title len.** | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 6 | 4 | 4 |

Table 5.4 Statistics of BIOMRC LARGE, LITE, TINY. The questions of the TINY version were answered by humans. All lengths are measured in tokens using a whitespace tokenizer.

candidate answers. Unlike BioRead, we use PUBTATOR [193], a repository that provides more than 30 million abstracts and their corresponding titles from PUBMED, with multiple annotations.[4] PUBTATOR used DNORM's biomedical entity annotations, which are more accurate than MetaMap's [92].

Following BioRead's methodology, we release two versions of BIOMRC, LARGE and LITE, containing 812$k$ and 100$k$ instances respectively, for researchers with more or fewer resources, along with the 60 instances (TINY) humans answered. Random samples from BIOMRC LARGE were selected to create LITE and TINY. BIOMRC TINY is used only as a test set for human evaluation; it has no training and validation subsets.

[4]Like PUBMED, PUBTATOR is supported by NCBI. Consult: `www.ncbi.nlm.nih.gov/research/pubtator/`

## 5.4 Reading Comprehension Models

We re-implemented and tested on BioReadLite and BIOMRC LITE two well-known MRC methods, AS-READER [77] and AOA-READER [37]. The authors of the AOA-READER models even though they reported state-of-the-art results on the BookTest dataset did not release the code to replicate their results. Therefore we re-implemented their deep learning MRC model along with the AS-READER model and made efforts to improve both of them. We test these two models along with four baselines, as a first step towards a leaderboard for models trained and tested on our introduced datasets. We also introduce two transformer-based deep-learning models for MRC which managed to surpass all baselines and competitive deep-learning models setting a new state-of-the-art for our BIOMRC dataset.[5] We made publicly available the re-implementations of the two MRC methods [6] and the code of the newly introduced MRC models.[7]

### 5.4.1 Attention Sum Reader (AS-READER)

**AS-READER** [77] uses a bidirectional recurrent neural network (biRNN) [163, 166] with GRU units [34] to process the passage (context) and another one to process the cloze-style question (Figure 5.1). The last states of the first biRNN (the concatenated last states of the two directions) represent the cloze-style question, whereas the states of the second biRNN (the concatenated states of the two directions, for each token position) are used as context-sensitive embeddings of the passage tokens. The dot product between the question representation and the context-sensitive embedding of each passage token is then computed, and a softmax is applied to the dot products to turn them into attention scores ranging from 0 to 1. The candidate answers can only be single tokens of the passage. If a candidate answer occurs multiple times in the passage, its attention scores are summed. Finally, the candidate answer with the largest (summed) attention score is selected.

### 5.4.2 Attention Over Attention Reader (AOA-READER)

**AOA-READER**'s architecture is presented in Figure 5.2. It uses a biRNN to create context-sensitive embeddings for each passage token, as in AS-READER. Another biRNN processes the question, but instead of keeping only the (concatenated) last states of the two directions as the question representation, all the states of the question biRNN (the concatenated states from

[5] We have only tested the new deep-learning models on BIOMRC since BIOMRC is an improved version of BIOREAD dataset.

[6] `https://github.com/dpappas/BIOREAD_code`

[7] `https://github.com/PetrosStav/BioMRC_code`

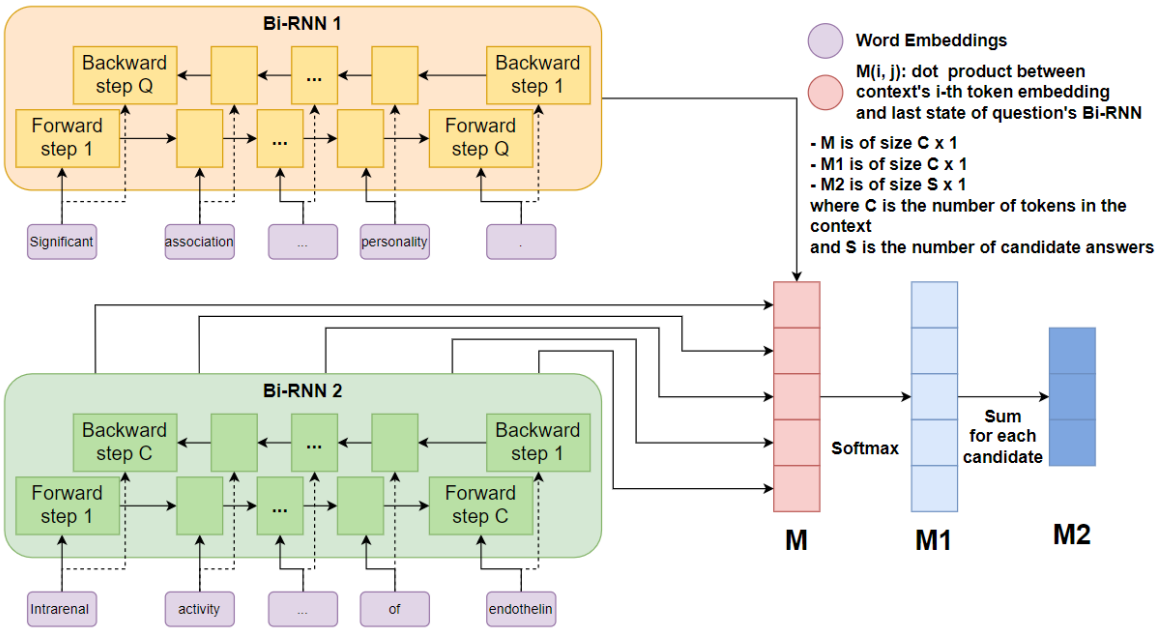


Fig. 5.1 Architecture of the AS-READER Model.

both directions, for each question token position) are kept as context-sensitive embeddings of the question tokens. The dot product between each context-sensitive embedding of the passage and each context-sensitive embedding of the question is then computed, leading to a matrix $M$ of dimensions $C \times Q$ of dot products, where $C$ and $Q$ are the lengths of the passage (context) and question, respectively, in tokens. Intuitively, each element $m_{i,j}$ of $M$ shows how relevant token $i$ of the passage is to token $j$ of the question. The $i$-th row of $M$ contains $Q$ scores, showing how relevant each token of the question is, from the viewpoint of the $i$-th token of the passage. The rows of $M$ are averaged (after applying a softmax to each row first) to obtain a single row-vector $CA$ with $Q$ scores that show how relevant each token of the question is with respect to *all* the tokens of the passage.

Similarly, the $j$-th column of $M$ contains $C$ scores showing how relevant each token of the passage is, from the viewpoint of the $j$-th token of the question. The matrix-vector multiplication $M1 \times CA^T$, where $M1$ is the original $M$ with a softmax applied to each column, produces $TS$ scores that show how important each passage token is from the viewpoint of the entire question, as captured by $CA$. A softmax is applied to the $TS$ scores, to turn them into attention scores from 0 to 1. As in AS-READER, the candidate answers can only be single tokens of the passage. If a candidate answer occurs multiple times in the passage, its attention scores are summed and form the $CS$ scores in Figure 5.2. Finally, the candidate answer with the largest (summed) attention score is selected.

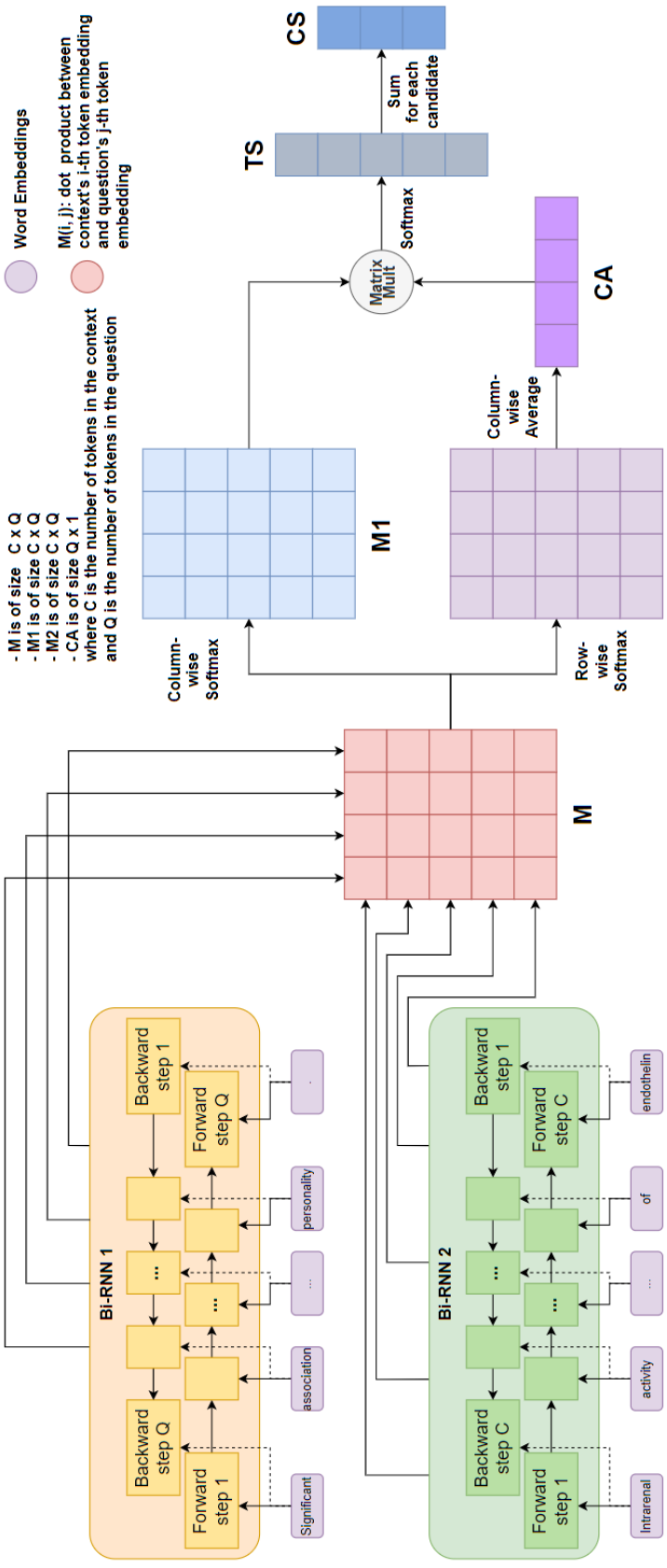


Fig. 5.2 Architecture of the AOA-READER Model.

### 5.4.3 Improvements on AS-READER and AOA-READER

We re-implemented the described AS-READER and AOA-READER models. We then experimented with improvements to the structures of the models. We observed that it is not necessary to apply the attention mechanism on tokens that are not candidate answers since the error back propagates only through the RNN for these tokens.

In Figure 5.2 the 'CS' vector stores the score of each candidate answer which is computed as the sum of the scores of all occurrences of the candidate in the passage. Since only the scores of the candidates are used to compute and back-propagate a loss, many values stored in M, M1, and M2 are not used, since only candidate scores are selected from the 'TS' vector. Therefore a first and straightforward improvement of these models was to average the emissions of the passage-RNN for each candidate before applying the attention mechanism (before computing the M matrix). We still used the contextual vector representations produced by the biRNNs but instead of using all contextual vector representations from the biRNN that consumes the passage, we only use the contextual vector representations that correspond to occurrences of candidate answers. Additionally, for each candidate, we average all vector representations produced by the biRNN resulting in one vector per candidate. Then, as in the original AOA-READER model, we compute the dot product between the candidates' averaged contextual vector representations and the questions' token vector representations to compute a matrix M which is smaller than the original implementation (M has as many rows as the total number of distinct candidates in the passage).

We also experimented with replacing the attention mechanism of the models (dot product between a question's token vector representation and a passage's token vector representation) using:

- a self-attention mechanism [168] applied on the contextual vector representations of the passage and the question tokens produced by the RNN,
- an MLP with trainable parameters which consumes the concatenation of the vector representation of a question's token and the vector representation of a passage's token.

Even though we did not improve the results of the models, we managed to produce faster versions of the AS-READER and AOA-READER models which also demand fewer computational resources since we reduced the size of the attention matrices. However, in all of our experiments with BioRead and BIOMRC, we use the structure of the models reported in their original papers.

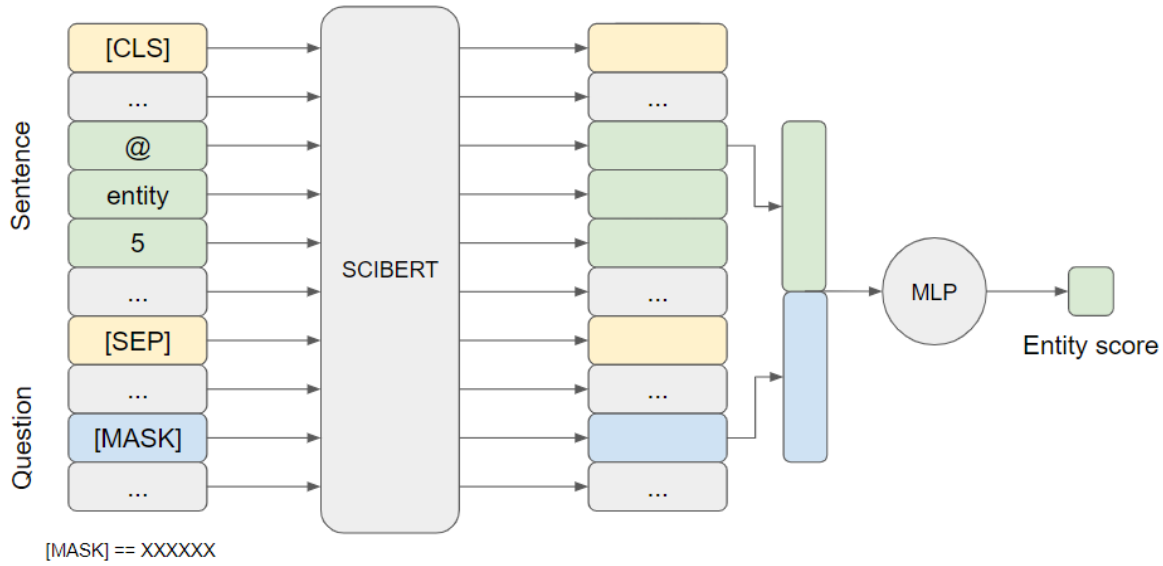


Fig. 5.3 Illustration of our SCIBERT-based models. Each sentence of the passage is concatenated with the question and fed to SCIBERT. The top-level embedding produced by SCIBERT for the first sub-token of each candidate answer is concatenated with the top-level embedding of [MASK] (which replaces the placeholder XXXX) of the question, and they are fed to an MLP, which produces the score of the candidate answer. In SCIBERT-SUM-READER, the scores of multiple occurrences of the same candidate are summed, whereas SCIBERT-MAX-READER takes their maximum.

### 5.4.4 BERT-SUM and BERT-MAX Reader

AS-READER when first introduced was the state-of-the-art model for cloze-style QA, and this was mainly due to the use of the attention mechanism between the question and the passage. A major advantage of the transformer-based models is the more elaborate attention mechanism that is applied to the input and their pre-training. Therefore we created BERT-based models for cloze-style QA and further trained (fine-tuned) them on the newly created datasets. We use SCIBERT [16], a pre-trained BERT [42] model for scientific text. SCIBERT is pre-trained on 1.14 million articles from Semantic Scholar,[8] of which 82% (935$k$) are biomedical while the rest of the articles belong to the computer science domain.

For each passage-question instance, we split the passage into sentences using NLTK [20]. We concatenate each sentence with the cloze-style question, after replacing the placeholder 'XXXX' with BERT's [MASK] token. Each sentence-question pair is then consumed by SCIBERT (Figure 5.3) and the top-level vector representations of the entity identifiers (@entity$N$) of the sentence are collected along with the vector representation of the [MASK] token. As BERT's tokenizer splits the entity identifiers into sub-word units [42] (see also section 2.4),

[8] https://www.semanticscholar.org/

we use the vector representation of the first sub-word to represent the entire entity. The top-level token representations of BERT are context-aware, and it is common to use the first or last sub-token of each named entity.

For each entity of the sentence, we concatenate its top-level representation with that of the [MASK] token, and we feed them to a Multi-Layer Perceptron (MLP) to obtain a score for the entire entity (candidate answer). Following the same process, we compute one score for any entity in the passage. If an entity occurs multiple times in the passage, we take the sum or the maximum of the scores of its occurrences. In both cases, a softmax is then applied to the scores of all the entities, and the entity with the maximum score is selected as the answer. We call this model SCIBERT-SUM-READER or SCIBERT-MAX-READER, depending on how it aggregates the scores of multiple occurrences of the same entity (Figure 5.4).

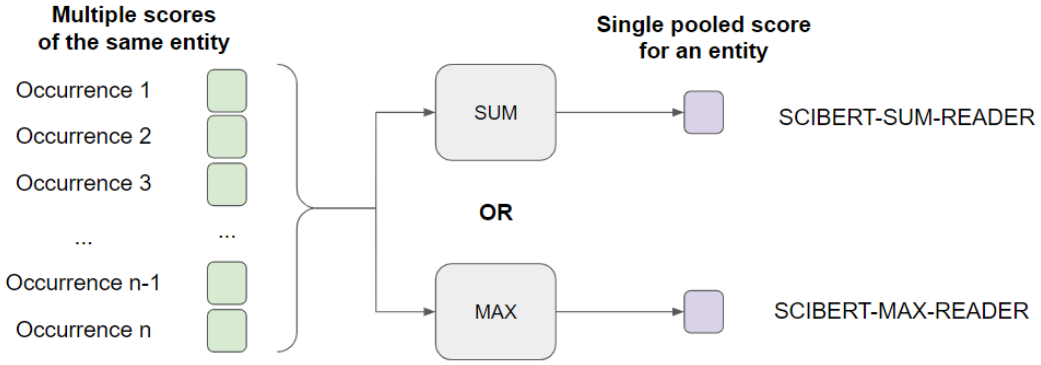


Fig. 5.4 Architecture of the SCIBERT-SUM-READER and the SCIBERT-MAX-READER.

### 5.4.5 Baselines

The first baseline (BASE1), returns the candidate answer (@entityID) that occurs most frequently in the context (passage), on the grounds that this candidate answer is more likely to have also occurred in the question (a sentence that follows the passage in the BioRead dataset or the title in the BIOMRC dataset) and, hence, more likely to have been converted to @placeholder.

The second and third baselines (BASE2 and BASE3), return the candidate answer that occurs first or last in the context, respectively. The last candidate answer is arguably more likely to be repeated in the question and, hence, more likely to have been converted to @placeholder, whereas the first candidate is the least likely to be repeated in that sense.[9] We also suspected that the biRNN encoder of the passage of AS-READER and AOA-READER

[9]These baselines were initially designed for BioRead and were then maintained in BIOMRC to allow comparing results.

would tend to 'remember' more the last (in the forward RNN) and the first (in the backwards RNN) tokens (and candidate answers) of the passage, in an extreme case behaving like BASE2 and BASE3.

In the fourth baseline, BASE4, we first extract all the token $n$-grams ($n = 2$) of the question that contain the @placeholder.[10] For each candidate answer (@entityID), we then replace the @placeholder in all the extracted $n$-grams by the particular candidate answer and count the total number of occurrences of the resulting $n$-grams in the context. The candidate answer with the largest total number of $n$-gram occurrences is returned as the answer.

In BioRead, the correct answer is never (by construction) the most frequent entity of the passage. Therefore, unless there are multiple entities with the same highest frequency, BASE3 performs poorly. Hence in BIOMRC experiments, we also include a variant, BASE3+, which randomly selects one of the entities of the passage with the same highest frequency, if multiple exist, otherwise, it selects the entity with the second highest frequency.

## 5.5 Experiments, Results & Analysis

### 5.5.1 BioRead

#### Human performance

To get a rough estimate of how easily humans can answer the questions of BioRead, we randomly selected 30 instances from BioRead's test subset and gave them to three human annotators, who had no biomedical background. The annotators were shown the context and question of each instance (as in Table 5.1, right) in a user interface that displayed @entityIDs as hyperlinks, and they were asked to select (click on) the correct candidate answer (@entityID) (Figure 5.5). When the annotators felt they were clueless (or very uncertain) about the correct answer, they could indicate this by clicking on a button, but they were instructed to select an answer when they felt it was probably the correct one, even if they were not entirely sure.

The mean accuracy of the three annotators was 68.01% (77.27%, 65.22%, 61.54% per annotator), counting only instances they answered (78.89% on average, 73.33%, 76.67%, 86.67% per annotator). The mean pairwise inter-annotator agreement, measured as Cohen's Kappa [35], was 68.57, considering only questions answered by both annotators in each pair. If not answering a question is treated as an additional candidate answer, the mean pairwise Kappa becomes 50.32.

[10] We experimented with $2 \leq n \leq 6$, and selected $n = 2$, which led to the best results on both the development sets of BioReadLite and BIOMRC LITE.

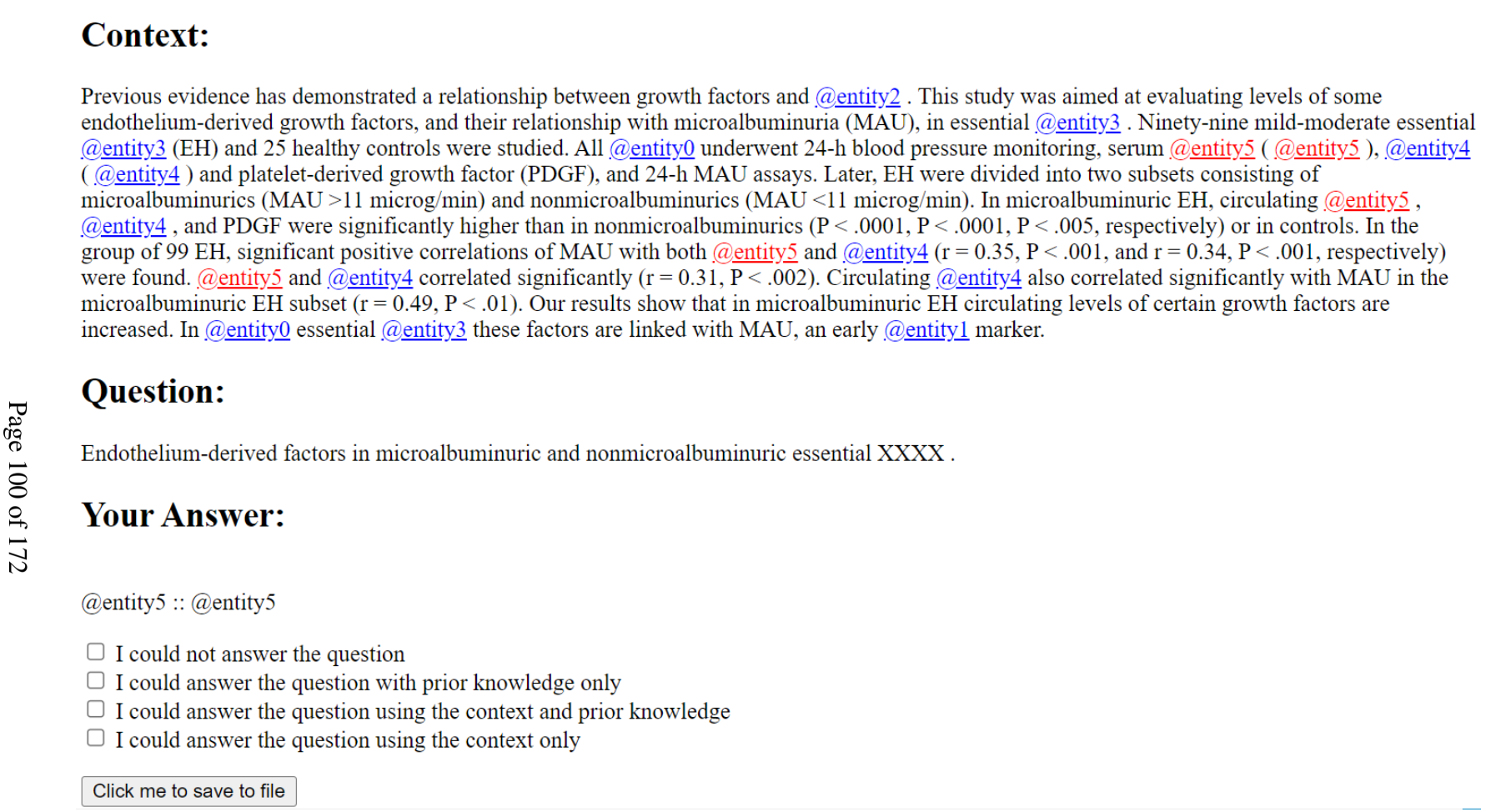


Fig. 5.5 Human annotation user interface for biomedical MRC. The users read the cloze-style question and select one entity from the context that replaces the placeholder ('XXXX' token) in the cloze-style question. When an entity is selected, all mentions of the same entity or synonyms to that entity are highlighted in the context. The users can then save their choice. The users can also provide feedback on whether they decided based only on the context, or prior knowledge or if they could not answer the question.

### Results on BioReadLite

Table 5.5 summarises our experimental results on BioReadLite; we did not have the computational resources to experiment with the full BioRead dataset, but we hope that others may be able to do so.[11] With the exception of the last row of Table 5.5, in all other cases we used setting A of Section 2, i.e., the identifier of each @entityID was unique only within the particular instance. For AS-READER and AOA-READER, we used the same hyper-parameter values as in the work of Kadlec et al. [78] and Cui et al. [37], respectively. Hence, a direct possible improvement would be to fine-tune the hyper-parameters for BioRead (or BioReadLite), which requires, however, substantial computational resources. We stopped training the two methods when their development loss had converged, i.e., after 5 epochs for AS-READER, 15 epochs for AOA-READER when using setting A, and 20 epochs for AOA-READER when using setting B; recall that in setting B the identifier of each @entityID is unique in the entire dataset. A single training epoch (including computing the development loss) takes 17, 21, and 22 hours, respectively. Performance is measured in terms of accuracy, i.e., number of correctly answered development or test instances, divided by the total number of development or test instances.

Table 5.5 shows that AOA-READER is clearly more accurate than AS-READER, at the expense of training speed, reaching 50.44% and 49.94% development and test accuracies with setting A, compared to 37.90% and 42.01% for AS-READER, respectively. These results confirm that the more elaborate attention mechanism of AOA-READER is important, as also reported in previous work [37, 11, 125]. Despite its simplicity, BASE1 (the most frequent candidate answer in the passage) is a reasonably strong baseline, reaching 26.86% development and 28.87% test accuracy, but AS-READER and AOA-READER outperform it by a wide margin. BASE2 and BASE3 are much weaker, suggesting that AS-READER and AOA-READER do not just remember the first or last candidate answers of the passage. The best baseline is BASE4 ($n$-grams). It scored 40.10% development and 37.20% test accuracy, surpassing AS-READER on the development subset, and challenging AS-READER on the test subset. Nevertheless, AOA-READER outperformed BASE4 by a wide margin. The performance of AOA-READER improved further (from 50.44% to 52.41% development accuracy, from 49.94% to 51.19% test accuracy), at the expense of additional training time, when setting B was used, i.e., when each entity ID was unique in the entire dataset, suggesting that AOA-READER was able to learn properties of at least some entities (concepts) from multiple training passages.

[11] We used a PC running Ubuntu, with 64 GB RAM, a 16 core CPU, and a GeForce GTX TITAN X GPU with 12GB memory.

We also trained the best method, AOA-READER with setting B, on smaller subsets of BioReadLite to study the effect of the size of the training set. We always used 20 epochs in this experiment, the number of epochs it took for the development loss to converge when using the entire training set of BioReadLite (Table 5.5, last row). Table 5.6 shows that increasing the size of the training set leads to improved development accuracy. We see a similar trend in test accuracy (from 49.22% to 51.19%) when going from 50% to 100% of the training set, but surprisingly the best test accuracy (51.51%) was obtained when using only 25% of the training set. The latter may be the result of a random fluctuation (e.g., the optimizer may have managed to find a better local minimum of the loss function in that case). It would be better to repeat each experiment multiple times, with different random parameter initializations, and report mean results (and standard deviations), but we did not have the required resources. Overall, however, it seems worth experimenting with the entire BioRead dataset, instead of BioReadLite, to see if its larger training subset would lead to significant improvements in accuracy.

| Method | Dev. Accuracy | Test Accuracy | Training Epochs |
|---|---|---|---|
| BASE1 set. A | 26.86 | 28.87 | n/a |
| BASE2 set. A | 8.14 | 9.38 | n/a |
| BASE3 set. A | 16.48 | 17.28 | n/a |
| BASE4 set. A | 40.10 | 37.20 | n/a |
| AS-READER set. A | 37.90 | 42.01 | $5 \times 17$ h |
| AOA-READER set. A | 50.44 | 49.94 | $15 \times 21$ h |
| **AOA-READER set. B** | **52.41** | **51.19** | **$20 \times 22$ h** |

Table 5.5 BioReadLite results (%), and the number of epochs (and time) required for the development loss to converge, when each entity ID is unique setting A in the particular instance only, or setting B in the entire dataset.

| Training Subset | Dev. Accuracy | Test Accuracy | Training Epochs |
|---|---|---|---|
| 25% | 47.06 | **51.52** | $20 \times 6$ h |
| 50% | 50.25 | 49.22 | $20 \times 11$ h |
| 100% | **52.41** | 51.19 | $20 \times 22$ h |

Table 5.6 BioReadLite results (%) of AOA-READER, with setting B, using the entire or only subsets of the training set.

### 5.5.2 BIOMRC

#### Human performance

We extended the human performance experiments applied on BIOREAD by asking both biomedical experts and non-biomedical experts to answer cloze-style questions. We also examined whether allowing annotators to see the original names of the biomedical entities would help them answer the question (maybe because of prior expert knowledge). We provided 30 questions (from BIOMRC LITE) to three non-experts (graduate CS students) in Setting A, and 30 other questions in Setting B. We also showed the same questions of each setting to two biomedical experts. In Setting A both the experts and non-experts were also provided with the original names of the biomedical entities (entity names before replacing them with @entity*N* pseudo-identifiers) to allow them to use prior knowledge; see the top three zones of Figure 5.6 for an example. By contrast, in Setting B the original names of the entities were hidden, to force the humans to base their answers on the contexts of the entities, not any knowledge about the entities themselves.

| | |
|---|---|
| Passage | The study enrolled 53 @entity1 (29 males, 24 females) with **@entity1576** aged 15-88 years. Most of them were 59 years of age and younger. In 1/3 of the @entity1 the diseases started with symptoms of @entity1729, in 2/3 of them–with pulmonary affection. @entity55 was diagnosed in 50 @entity1 (94.3%), acute @entity3617 –in 3 @entity1. ECG changes were registered in about half of the examinees who had no cardiac complaints. 25 of them had alterations in the end part of the ventricular ECG complex; rhythm and conduction disturbances occurred rarely. Mycoplasmosis @entity1 suffering from @entity741 ( @entity741 ) had stable ECG changes while in those free of @entity741 the changes were short. @entity296 foci were absent. @entity299 comparison in @entity1 with **@entity1576** and in other @entity1729 has found that cardiovascular system suffers less in acute mycoplasmosis. These data are useful in differential diagnosis of @entity296 . |
| Candidates | @entity1 : [‘patients’] ; **@entity1576** : [‘respiratory mycoplasmosis’] ; @entity1729 : [‘acute respiratory infections’, ‘acute respiratory viral infection’] ; @entity55 : [‘Pneumonia’] ; @entity3617 : [‘bronchitis’] ; @entity741 : [‘IHD’, ‘ischemic heart disease’] ; @entity296 : [‘myocardial infections’, ‘Myocardial necrosis’] ; @entity299 : [‘Cardiac damage’] . |
| Question | Cardio-vascular system condition in XXXX . |
| Expert Human Answers | annotator1: **@entity1576**; annotator2: **@entity1576**. |
| Non-expert Human Answers | annotator1: @entity296; annotator2: @entity296; annotator3: **@entity1576**. |
| Systems’ Answers | AS-READER: @entity1729; AOA-READER: @entity296; SCIBERT-SUM-READER: **@entity1576**. |

Fig. 5.6 Example from BIOMRC TINY. In Setting A, humans see both the pseudo-identifiers (@entity*N*) and the original names of the biomedical entities (shown in square brackets). Systems see only the pseudo-identifiers, but the pseudo-identifiers have global scope over all instances, which allows the systems, at least in principle, to learn entity properties from the entire training set. In Setting B, humans no longer see the original names of the entities, and systems see only the pseudo-identifiers with local scope (numbering reset per passage-question instance).

Table 5.7 reports the human and system accuracy scores on BIOMRC TINY. Both experts and non-experts perform better in Setting A, where they can use prior knowledge about the biomedical entities. The gap between experts and non-experts is three points larger in Setting B than in Setting A, presumably because experts can better deduce properties of the entities from the local context. Turning to the system scores, SCIBERT-MAX-READER is the best system, but much of its performance is due to the max-aggregation of the scores of multiple occurrences of entities. With sum-aggregation, SCIBERT-SUM-READER obtains exactly the same scores as AOA-READER, which again performs better than AS-READER. (AOA-READER and SCIBERT-SUM-READER make different mistakes, but their scores just happen to be identical because of the small size of TINY.) Unlike our results on BIOMRC LITE, we now see all systems performing better in Setting A compared to Setting B, which suggests they do benefit from the global scope of entity identifiers. Also, SCIBERT-MAX-READER performs better than both experts and non-experts in Setting A, and better than non-experts in Setting B. However, BIOMRC TINY contains only 30 instances in each setting, and hence the results of Table 5.7 are less reliable than those from BIOMRC LITE (Table 5.9).

| Method | Setting A | Setting B |
|---|---|---|
| **Experts (Avg)** | **85.00** | **61.67** |
| **Non-Experts (Avg)** | 81.67 | 55.56 |
| **AS-READER** (previous method) | 66.67 | 46.67 |
| **AOA-READER** (previous method) | 70.00 | 56.67 |
| **SCIBERT-SUM-READER** (new method) | 70.00 | 56.67 |
| **SCIBERT-MAX-READER** (new method) | **90.00** | **60.00** |

Table 5.7 Accuracy (%) on BIOMRC TINY. Best human and system scores are shown in bold.

#### Results on BIOMRC LITE

Table 5.9 reports the accuracy of all methods on BIOMRC LITE for Settings A and B. In both settings, all the neural models clearly outperform all the basic baselines, with BASE3 (the most frequent entity of the passage) performing worst and BASE3+ performing much better, as expected. In both settings, SCIBERT-MAX-READER clearly outperforms all the other methods on both the development and test sets. The performance of SCIBERT-SUM-READER is approximately ten percentage points worse than SCIBERT-MAX-READER's on the development and test sets of both settings, indicating that the superior results of SCIBERT-MAX-READER are to a large extent due to the different aggregation function (max instead of sum) it uses to combine the scores of multiple occurrences of a candidate answer, not to the extensive pre-training of SCIBERT. AOA-READER, which does not employ any pre-training,

| Annotators (Setting) | Kappa |
|---|---|
| **Experts setting A** | 70.23 |
| **Non Experts setting A** | 65.61 |
| **Experts setting B** | 72.30 |
| **Non Experts setting B** | 47.22 |

Table 5.8 Human agreement (Cohen's Kappa, %) on BIOMRC TINY. Avg. pairwise scores for non-experts.

is competitive to SCIBERT-SUM-READER in Setting A and performs better than SCIBERT-SUM-READER in Setting B, which again casts doubts on the value of SCIBERT's extensive pre-training. We expect, however, that the performance of the SCIBERT-based models, could be improved further by fine-tuning SCIBERT's parameters.

The performance of SCIBERT-SUM-READER is slightly better in Setting A than in Setting B, which might suggest that the model manages to capture global properties of the entity pseudo-identifiers from the entire training set. However, the performance of SCIBERT-MAX-READER is almost the same across the two settings, which contradicts the previous hypothesis. Furthermore, the development and test performance of AS-READER and AOA-READER is higher in Setting B than A, indicating that these two models do not capture global properties of entities well, performing better when forced to consider only the information of the particular passage-question instance. Overall, we see no strong evidence that the models we considered are able to learn global properties of the entities.

In both Settings A and B, AOA-READER performs better than AS-READER, which was expected since it uses a more elaborate attention mechanism, at the expense of taking longer to train (Table 5.9).[12] The two SCIBERT-based models are also competitive in terms of training time, because we only train the MLP (154k parameters) on top of SCIBERT, keeping the parameters of SCIBERT frozen.

[12]We trained all models for a maximum of 40 epochs, using early stopping on the dev. set, with patience of 3 epochs.

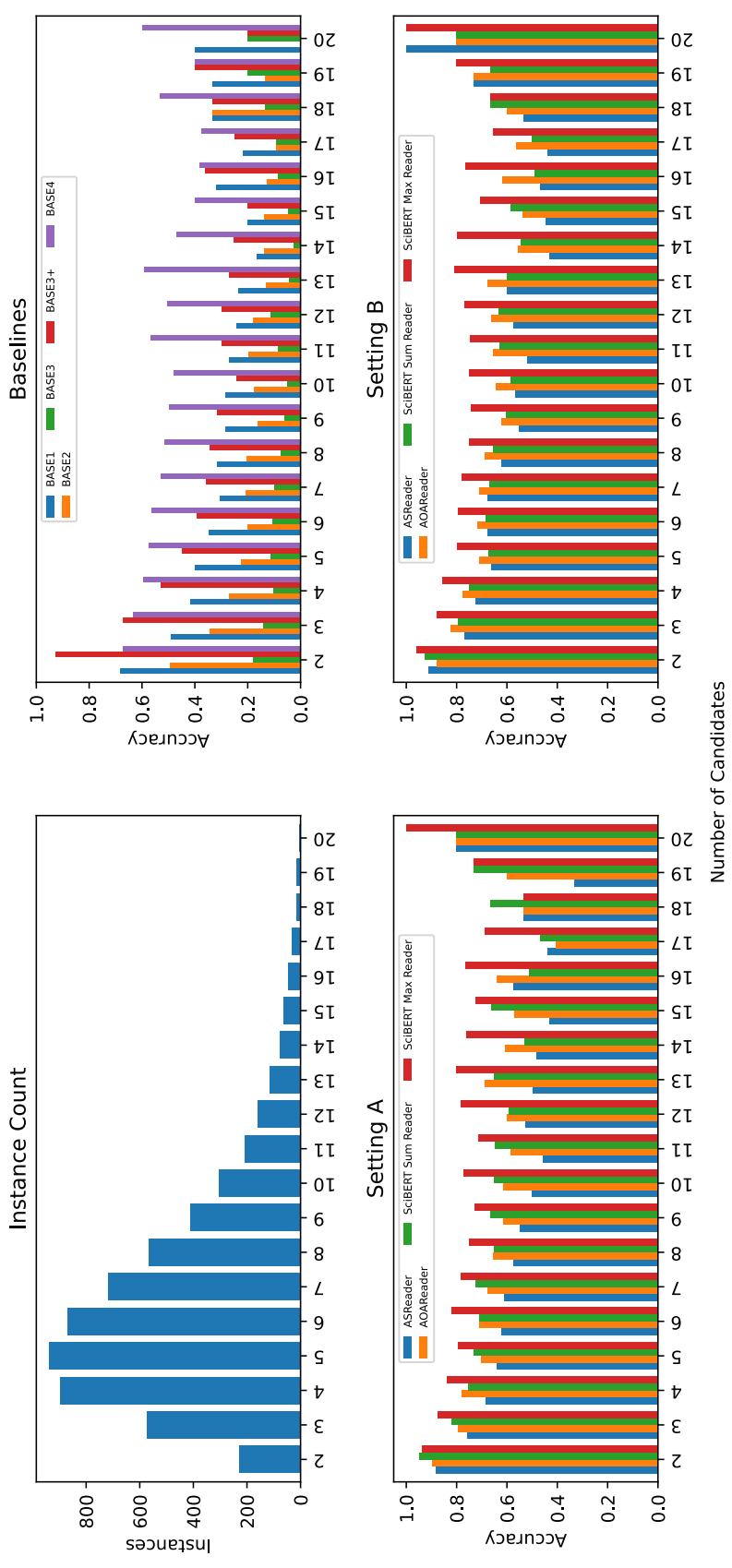


Fig. 5.7 More detailed statistics and results on the development subset of BIOMRC LITE. Number of passage-question instances with 2, 3, . . . , 20 candidate answers (top left). Accuracy (%) of the basic baselines (top right). Accuracy (%) of the neural models in Settings A (bottom left) and B (bottom right).

The trainable parameters of AS-READER and AOA-READER are almost double in Setting A compared to Setting B. To some extent, this difference is due to the fact that for both models we learn a word embedding for each @entity*N* pseudo-identifier, and in Setting A the numbering of the identifiers is not reset for each passage-question instance, leading to many more pseudo-identifiers (31.77k pseudo-identifiers in the vocabulary of Setting A vs. only 20 in Setting B); this accounts for a difference of 1.59M parameters.[13] The rest of the difference in total parameters (from Setting A to B) is due to the fact that we tuned the hyper-parameters of each model separately for each setting (A, B), on the corresponding development set. Hyper-parameter tuning was performed separately for each model in each setting, but led to the same numbers of trainable parameters for AS-READER and AOA-READER, because the trainable parameters are dominated by the parameters of the word embeddings. Note that the hyper-parameters of the two SCIBERT-based models (of their MLPs) were very minimally tuned, hence these models may perform even better with more extensive tuning.

AOA-READER was also better than AS-READER in our experiments on BioReadLite dataset, but the development and test accuracy of AOA-READER in Setting A of BioRead was 52.41% and 51.19%, respectively while in Setting B, it was 50.44% and 49.94%, respectively (see Table 5.5). The much higher scores of AOA-READER (and AS-READER) on BIOMRC LITE are an indication that the new dataset is less noisy, or that the task is at least more feasible for machines. Our results on BioReadLite were slightly higher in Setting A than in Setting B, suggesting that AOA-READER was able to benefit from the global scope of entity identifiers, unlike our findings in BIOMRC.

Figure 5.7 shows how many passage-question instances of the development subset of BIOMRC LITE have 2, 3, ..., 20 candidate answers (top left), and the corresponding accuracy of the basic baselines (top right), and the neural models ((bottom) in Settings A (bottom left) and B (bottom right)). BASE3+ is the best basic baseline for 2 and 3 candidates, and for 2 candidates it is competitive to the neural models. Overall, however, BASE4 is clearly the best non-neural baseline, but it is outperformed by all neural models in almost all cases, as in Table 5.9. SCIBERT-MAX-READER is again the best system in both settings, almost always outperforming the other systems. AS-READER is the worst neural model in almost all cases. AOA-READER is competitive to SCIBERT-SUM-READER in Setting A, and slightly better overall than SCIBERT-SUM-READER in Setting B, as can be also seen in Table 5.9.

[13] Hyper-parameter tuning led to 50- and 30-dimensional word embeddings in Settings A, B, respectively. AS-READER and AOA-READER learn word embeddings from the training set, without using pre-trained embeddings.

| | | BIOMRC Lite – Setting A | | | | | | | BIOMRC Lite – Setting B | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Method** | **Train Acc** | **Dev Acc** | **Test Acc** | **Train Time** | **All Params** | **Word Embed.** | **Entity Embed.** | **Train Acc** | **Dev Acc** | **Test Acc** | **Train Time** | **All Params** | **Word Embed.** | **Entity Embed.** |
| | **BASE1** | 37.58 | 36.38 | 37.63 | 0 | 0 | 0 | 0 | 37.58 | 36.38 | 37.63 | 0 | 0 | 0 | 0 |
| | **BASE2** | 22.50 | 23.10 | 21.73 | 0 | 0 | 0 | 0 | 22.50 | 23.10 | 21.73 | 0 | 0 | 0 | 0 |
| | **BASE3** | 10.03 | 10.02 | 10.53 | 0 | 0 | 0 | 0 | 10.03 | 10.02 | 10.53 | 0 | 0 | 0 | 0 |
| | **BASE3+** | 44.05 | 43.28 | 44.29 | 0 | 0 | 0 | 0 | 44.05 | 43.28 | 44.29 | 0 | 0 | 0 | 0 |
| | **BASE4** | 56.48 | 57.36 | 56.50 | 0 | 0 | 0 | 0 | 56.48 | 57.36 | 56.50 | 0 | 0 | 0 | 0 |
| **AS-READER** | (previous method) | **84.63** | 62.29 | 62.38 | 18 x 0.92 hr | 12.87M | 12.69M | 1.59M | 79.64 | 66.19 | 66.19 | 18 x 0.65 hr | 6.82M | 6.66M | 0.60k |
| **AOA-READER** | (previous method) | 82.51 | 70.00 | 69.87 | 29 x 2.10 hr | 12.87M | 12.69M | 1.59M | **84.62** | 71.63 | 71.57 | 36 x 1.82 hr | 6.82M | 6.66M | 0.60k |
| **SCIBERT-SUM-READER** | (new method) | 71.74 | 71.73 | 71.28 | 11 x 4.38 hr | **154k** | **0** | **0** | 68.92 | 68.64 | 68.24 | 6 x 4.38 hr | **154k** | **0** | **0** |
| **SCIBERT-MAX-READER** | (new method) | 81.38 | **80.06** | **79.97** | 19 x 4.38 hr | **154k** | **0** | **0** | 81.43 | **80.21** | **79.10** | 15 x 4.38 hr | **154k** | **0** | **0** |

Table 5.9 Training, development, test accuracy (%) on BIOMRC LITE in Settings A (global scope of entity identifiers) and B (local scope), training times (epochs $\times$ time per epoch), and the number of trainable parameters (total, word embedding parameters, entity identifier embedding parameters). SCIBERT's parameters are not fine-tuned. In the lower zone (neural methods), the difference from each accuracy score to the next best is statistically significant ($p < 0.02$). We used single-tailed Approximate Randomization [48], randomly swapping the answers to 50% of the questions for 10k iterations.

## 5.6 Conclusions and Future work

In this work, we made significant contributions to the field of machine reading comprehension in the biomedical domain. Firstly, we introduced two large datasets that provide ample opportunities for the development and testing of new machine reading comprehension models. Secondly, we introduced two deep learning models that are specifically designed for this task. Our models were thoroughly evaluated against strong baselines, and they surpassed all of them. Finally, we conducted a comparison against human annotators, and while our models outperformed non-expert humans, they were unable to surpass human annotators with specialized biomedical expertise. These results indicate that there is still room for improvement in this field and that future work will be necessary to further advance the state of the art.

Future work could tune more extensively the BERT-based model to further improve its efficiency, and investigate if some of its techniques (mostly its max-aggregation, but also using sub-tokens) can also benefit the other neural models we considered (AS-READER and AOA-READER). One could also experiment with other MRC models that recently performed particularly well on open-domain MRC datasets [224]. Finally, one could explore if pre-training neural models on BIOREAD is beneficial in human-generated biomedical datasets [184].

The BIOREAD and BIOMRC datasets, being cloze-type automatically generated datasets, don't have the complexity of a real-world (not-synthetic) Question Answering [96] dataset like SQUAD [154]. However, we believe that we could use our datasets to pretrain reading comprehension models before fine-tuning for more complex Biomedical Question Answering datasets. Alternatively, artificial datasets like BIOREAD and BIOMRC can be used for data augmentation, and this direction is explored in the next chapter.

# Chapter 6

# Biomedical Factoid QA and Data Augmentation

## 6.1 Introduction

Training supervised machine learning models requires a substantial amount of annotated data. Data augmentation (DA) techniques are widely used to increase robustness and improve results on several NLP tasks when there is a limited number of training data.

Data augmentation in NLP refers to the process of artificially increasing the size of a dataset by generating new samples from the existing data. This is done to improve the performance of a model by increasing the diversity of the training data. The new samples are created by applying various techniques such as changing the order of words, adding or removing words, replacing words with synonyms or random words, or other techniques such as the ones discussed in this chapter. The goal of data augmentation is to make the model robust to various forms of input, so it can generalize better to unseen data and handle real-world variations of the data. This is particularly important in NLP, where small datasets and limited training data are often a challenge, making data augmentation an important technique for improving the performance of NLP models. While several techniques for data augmentation are widely used to train deep neural networks for computer vision, no data augmentation technique has yet emerged as a standard practice for data augmentation in NLP.

In this chapter, we investigate the effectiveness of seven data augmentation techniques (namely using Machine Reading Comprehension data, increasing the context of the text, Back Translation, Question Generation, Information Retrieval, and two token substitution techniques) on training deep neural networks for biomedical factoid QA, using BIOASQ-8 (2021), Phase B, Task B data (see also Section 3.2). To our knowledge, we are the first to

ever compare seven data augmentation techniques in QA. BIOASQ contains only a limited number of factoid question-answer (QA) pairs that have been annotated by human experts. This small amount of annotated data makes it challenging for QA models to generalize to unseen questions. We show that using any of the proposed data augmentation techniques improves results on BIOASQ factoid QA.

We also experiment with COVIDQA [121], a dataset containing 2,019 question/answer pairs annotated by volunteer biomedical experts on scientific articles related to COVID-19. Using the data augmentation techniques that performed best on BIOASQ data, we also improved the results of a deep-leaning factoid QA model for the COVIDQA dataset.

## 6.2 Related Work

DA is a key ingredient of success in deep learning for computer vision [171]. DA for NLP has been explored less, but is an active research area [172, 53], with methods ranging from leveraging knowledge graphs [124] to generating textual data from scratch [208, 14]. The most common NLP task in DA is text classification [13]. Feng et al. [53] consider span-based NLP tasks in specialized domains, which includes biomedical MRC, among the most challenging for DA.

Word substitution is a simple and common DA approach in NLP. In thesaurus-based substitution [76, 2], words are replaced by synonyms or closely related words (e.g., hypernyms). Word embedding substitution [192] replaces words by others nearby in a pre-trained vector space model (see Section 6.3.2). Alternatively, a random word is removed, inserted [194, 117], or noised with spelling errors [15]. Sentences may also be re-ordered or removed [169, 32]. Text generation has also been used in several NLP tasks for adversarial augmentation [33], to paraphrase training examples [155, 27, 196], or generate new [9, 85]. Back-translation [164] is also widely used across NLP tasks [172, 53](see Section 6.3.2).

Previous DA work for *QA in particular* includes back-translation [49], question generation [221, 8, 28, 108, 208], paraphrasing [47, 104], and synonym replacement [134], but not in a biomedical setting. The IR-based DA we used (Section 6.3.2 bellow) follows Yang et al. [205], who experimented in English and Chinese, but not in the biomedical domain. Expanding the passage with surrounding sentences (Section 6.3.2 bellow) follows Yoon et al. [213], who used this method in BIOASQ. Dhingra et al. [44] create artificial cloze-style MRC datasets (similar to the ones we created in Chapter 4, but not biomedical) and use them to pre-train neural QA models (but not Transformer-based), which are then fine-tuned on real training examples. By contrast, we use artificial MRC datasets to fine-tune large pre-trained Transformers. All the above studies experimentally compare at most two DA methods; we

compare seven. Hence, our work is the largest (in terms of methods considered) experimental study of DA for QA (and possibly NLP).

Longpre et al. [105] report that back-translation did not improve generalization in (non-biomedical) QA experiments with fine-tuned pre-trained Transformers, unlike our findings. Longpre et al. [106] report that back-translation and Easy Data Augmentation [194] are not always effective in text classification when fine-tuning pre-trained Transformers, even with small end-task training sets. Consequently, Feng et al. [53] recommend exploring when DA is effective for large pre-trained models. Our work contributes to this discussion by showing that DA can lead to very substantial performance gains, even when using large pre-trained Transformers fine-tuned on large generic (SQUAD) and/or small domain-specific (BIOASQ) end-task datasets.

## 6.3 Experimental Setup

### 6.3.1 Model

In our experiments, we use pre-trained deep learning models available from Hugging Face.[1] Huggingface offers notebooks of open-source code with which most users train their own question answering deep learning models.[2] The provided code however trains a DL model which extracts a maximum number of one answer span in the given text computing only a single start offset and a single end offset.

We slightly altered Hugging Face's code to allow the extraction of multiple answers in the text. The original code is fed with a question concatenated with a text that contains the answer. The input is split to BPEs (see Section 2.4) and a distilled BERT model extracts a vector representation for each BPE (In our experiments, we use an ALBERT-XL model instead of DISTILBERT and justify our decision in Section 6.3.2 below). Then an MLP is used to extract two logits for each BPE as shown in Figure 6.1. The two logits are used as probability scores of the BPE being the start of the answer span or the end of the answer span respectively. To train the model, a softmax activation function is used across the start logits of all BPEs and a second softmax activation function across the end logits of all BPEs. A categorical cross-entropy loss is applied in both cases (start logits, end logits) per token and an overall loss is computed.

[1] Consult `https://huggingface.co/models`

[2] The code uses SQUAD as a corpus and can be found in the following link: `https://colab.research.google.com/github/huggingface/notebooks/blob/master/examples/question_answering.ipynb`

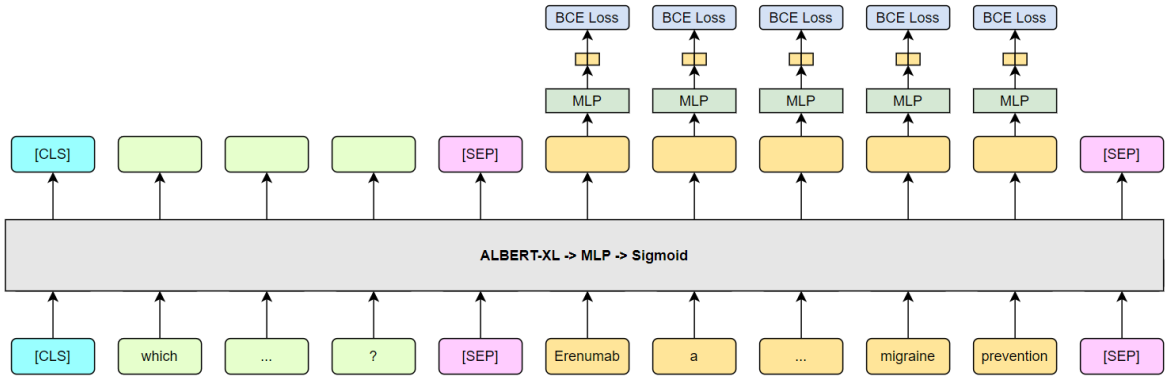


Fig. 6.1 Architecture of the model used for all of the following factoid QA experiments. ALBERT-XL is fed with a question and snippet. Its contextualized embeddings are passed through an MLP with SIGMOID activations that produces a begin ($P_b$) and end ($P_e$) probability per token of the snippet.

In our dataset, there are often multiple occurrences of the answer in the input of the models. Softmax is a soft version of argmax that tries to assign a high probability score to only one of the logits (in the same that an argmax returns 1 for only one of the positions of the vector it is fed with). In the case where multiple occurrences of the answer are found in the text, there are multiple tokens that are beginnings (or ends) of answer spans.

One way to represent the gold truth labels for a case with two occurrences would be to split the probability mass (0.5 for each occurrence), however categorical cross-entropy cannot handle these cases. Another one would be to create two training instances where each instance would have the value of 1 to different positions. In the latter scenario, the model would be trained to compute a different output given the same input.

In our version of the code, we replace the output softmax activation function with the SIGMOID function. Additionally, instead of using categorical cross-entropy loss we compute the binary cross-entropy loss between each logit and the gold truth per token, and we sum the (per-token) binary cross-entropy losses.

### 6.3.2 Data Augmentation Approaches

#### Off-the-shelf models

As a starting point, we compared the performance of three publicly available pre-trained models that have already been fine-tuned for MRC on SQUAD [154] or SQUAD-V2 [153].[3]

[3]We obtained the models from `https://huggingface.co/ktrapeznikov/albert-xlarge-v2-squad-v2`. We use the XL version of ALBERT. The other two models adopt the BERT-BASE architecture; no XL variants were available.

At the time of our experiments, ALBERT-based models [91] were among the best on the SQUAD leaderboards; here, we use ALBERT fine-tuned on SQUAD-V2. We also considered BIOBERT [94], because it is pre-trained on a biomedical corpus; again, we use it fine-tuned on SQUAD-V2. The third model, DISTILBERT [162], was chosen because of its much smaller size, which makes running experiments easier. This model is pre-trained on a generic corpus, like the original BERT, and we use it fine-tuned on SQUAD. All three models are used here off-the-self, i.e., they are only evaluated, not trained in any way on BIOASQ data. Throughout this work, we use the development subset of the BIOASQ data to select models and configurations of DA methods, but in this particular experiment we use the union of the training and development subsets, since no BIOASQ training is involved. ALBERT is the best off-the-shelf model considered (Table 6.1), hence we use it in all other experiments.

| Model | PRAUC (BIOASQ train+dev) |
|---|---|
| DISTILBERT (SQUAD) | 64.27 |
| BIOBERT (SQUAD-V2) | 69.22 |
| **ALBERT (SQUAD-V2)** | **75.05** |

Table 6.1 *Off-the-shelf* pre-trained models, fine-tuned for MRC on SQUAD or SQUAD-V2. We report Precision-Recall AUC (PRAUC, %) on BIOASQ training and development data, since no BIOASQ training is involved.

### Baselines

We use two baselines that do not involve DA: i) off-the-shelf ALBERT, pre-trained on a generic corpus, already fine-tuned on SQUAD-V2 (last model of Table 6.1); and ii) same as the first baseline, but further fine-tuned (on top of the fine-tuning on SQUAD-V2) on our BIOASQ training data, with the modified architecture of Section 6.3.1. Table 6.2 shows that the second baseline is much stronger. Hence, we report performance gains with DA methods against the second baseline in subsequent sections.[4]

### Settings

In our experiments, we examine three settings for training deep learning models with augmented data. In setting PRETRAIN we only use augmented data without using any training instances of the original BIOASQ data and report the results. If the DA techniques produce data of poor quality then results on PRETRAIN setting will be worse than using

[4]We also experimented with pre-trained ALBERT directly fine-tuned only on BIOASQ, but the performance was much worse.

| Model | +train ex. | PRAUC (BIOASQ dev) |
|---|---|---|
| ALBERT (SQUAD-V2) | 0 | 80.25 |
| +BIOASQ | 2,848 | 89.57 |

Table 6.2 Performance of baselines on BIOASQ development data. The first one is ALBERT-XL fine-tuned on SQUAD-V2. The second one is further fine-tuned on BIOASQ, with the modified architecture of Figure 6.1. We also show the number of domain-specific (BIOASQ) training examples.

the original BIOASQ data. In setting FINETUNE we further train the model that produces the best development score in the PRETRAIN setting with the original BIOASQ data. In this setting, the model consumes the original data after the augmented data therefore it may correct noise introduced by augmented data. In setting COMB we shuffle the original data and the augmented data and produce one training set. In our experiments, we examine all three settings but only report the results of the best setting in each experiment. A detailed version of all the results (FINETUNE, PRETRAIN, COMB) can be found in the Appendix.

#### BIOMRC

For this augmentation method, we use BIOMRC [145], which is described in Chapter 5 and is one of the most recent and largest *artificial* cloze-style biomedical MRC datasets. From the two versions of the BIOMRC dataset, we chose to use BIOMRC LITE which includes 100*k* cloze-style questions. Each 'question' is the title of a biomedical article, with an entity mentioned in the title hidden. Each question is accompanied by a passage (the abstract of the article), candidate answers (entities mentioned in the abstract), and the gold answer. From each passage, we keep only the sentence containing the gold answer as the given snippet, and we generate a question-snippet-answer triple. If more than one sentences of the passage contain the gold answer, we create multiple triples, one for each sentence. We end up with approximately 142*k* artificial training triples.

When BIOMRC is used for data augmentation the results improve and achieve 93.15 dev PRAUC for the development set (Table 6.4). Even if we only use the BIOMRC data to train our models, 10*k* training examples suffice to surpass the results of the original BIOASQ dataset. This once more confirms the value of the BIOMRC dataset. Better results can be obtained if the entire BIOMRC dataset (instead of the BIOMRC LITE) is used to train our model.

| DA with instances from BIOMRC | |
|---|---|
| ID | Instance |
| 19823942 | **BIOMRC question:** Systolic versus diastolic cardiac function variables during [MASK] treatment for breast cancer .<br>**BIOMRC snippet:** epirubicin induces considerable decrease in left ventricular ejection fraction and a high risk of CHF.<br>**BIOMRC answer:** epirubicin |

Table 6.3 A training instance extracted from BIOMRC. Each instance is a triple containing a cloze-style question, a snippet, and the span of the snippet answering the question. We have underlined the answer span found in the snippet.

| ALBERT (SQUAD-V2) | +train ex. | PRAUC (BIOASQ dev) |
|---|---|---|
| +BIOASQ | 2,848 | 89.57 |
| +BIOMRC | 2,848 | 78.66 |
| +BIOMRC +BIOASQ | 5,696 | 91.57 |
| +BIOMRC | 10,000 | 91.20 |
| +BIOMRC +BIOASQ | 12,848 | **93.15** |
| +BIOMRC | 30,000 | 90.57 |
| +BIOMRC +BIOASQ | 32,848 | 92.19 |
| +BIOMRC | 50,000 | 91.19 |
| +BIOMRC +BIOASQ | 52,848 | 91.51 |
| +BIOMRC | 100,000 | 90.93 |
| +BIOMRC +BIOASQ | 102,848 | 92.39 |

Table 6.4 Adding training examples from an *artificial cloze-style* MRC dataset (BIOMRC). The '+train ex.' column shows the number of domain-specific training examples (from BIOMRC and/or BIOASQ) used, on top of examples seen during fine-tuning on SQUAD-V2.

### Token Substitution

In token substitution, we use pre-trained biomedical language models to replace tokens of the input with words extracted from the language models. In our first approach for token substitution, we use biomedical WORD2VEC [119, 24] embeddings. Given a question-snippet-answer training instance, we consider all the word tokens of the snippet (excluding stop-words). For each token $w_i$ ($i = 1, \ldots, n$) of the snippet, we select the $k_i \leq K$ most similar words $w_j$ ($j = 1, \ldots, k_i$) of the vocabulary, using cosine similarity of word embeddings $(\vec{w}_i, \vec{w}_j)$, that satisfy $\cos(\vec{w}_i, \vec{w}_j) \geq C$. We then produce $(k_1 + 1)(k_2 + 1)\ldots(k_n + 1) - 1$ artificial training instances by replacing each token $w_i$ of the snippet with one of its $k_i$ most similar words (or itself), requiring at least one token of the original snippet to have been replaced. We then randomly sample $10k$ to $100k$ of the resulting instances and use them as

additional training examples. We set $K = 10$, $C = 0.95$ based on preliminary experiments on development data. Adding $10k$ of the resulting artificial training examples to the original BIOASQ examples leads to 95.60 development PRAUC, outperforming the strong baseline (89.57) by six percentage points (Table 6.6). Using only the $10k$ artificial examples, without any of the original examples, achieves almost identical performance (95.59), which suggests that the generated examples are of high quality. As when using artificial MRC examples (Table 6.4), adding more than $10k$ artificial instances provides no further benefit, probably because we end up adding too many minor variants of the same original examples.

| **DA with word substitution based on WORD2VEC** | |
|---|---|
| ID | Instance |
| 21546092 | **BIOASQ snippet:** Beck's Medical Lethality Scale (BMLS) was administered to assess the degree of medical injury, and the SAD PERSONS mnemonic scale was used to evaluate suicide risk.<br>**BIOASQ question:** What is evaluated with the SAD PERSONS scale?<br>**Snippet after WORD2VEC substitution:** becks medical lethality scale bmls was administered to evaluate the degree of medical injury and the sad people domain-general scale was utilized to investigate suicide risk<br>**BIOASQ answer:** suicide risk |

Table 6.5 A training instance generated via word substitution based on WORD2VEC. We randomly select at most 10 words of a BIOASQ snippet and substitute each word $w_i$ with its most similar word $w_j$ from the vocabulary of the WORD2VEC model. Highlights of the same color indicate substituted words and the corresponding substitutions.

| **ALBERT (SQUAD-V2)** | **+train ex.** | **PRAUC (BIOASQ dev)** |
|---|---|---|
| + BIOASQ | 2,848 | 89.57 |
| +WORD2VEC | 2,848 | 95.56 |
| +WORD2VEC +BIOASQ | 5,696 | 95.27 |
| +WORD2VEC | 10,000 | 95.59 |
| +WORD2VEC +BIOASQ | 12,848 | **95.60** |
| +WORD2VEC | 30,000 | 95.28 |
| +WORD2VEC +BIOASQ | 32,848 | 95.20 |
| +WORD2VEC | 50,000 | 95.16 |
| +WORD2VEC +BIOASQ | 52,848 | 95.13 |
| +WORD2VEC | 100,000 | 95.36 |
| +WORD2VEC +BIOASQ | 102,848 | 95.22 |

Table 6.6 Data augmentation with WORD2VEC-*based word substitution*, using biomedical embeddings.

In the second approach for token substitution, we use BIOLM [99] and specifically a ROBERTA -LARGE model pre-trained on PUBMED, PMC, and MIMIC-III [226] with a new vocabulary extracted from PUBMED.[5] We use the same process as in WORD2VEC word substitution, but each candidate replacement $w_j$ of an original word $w_i$ of the snippet must now satisfy $p(w_j) \geq P$ (instead of $\cos(\vec{w}_i, \vec{w}_j) \geq C$), where $p(w_j)$ is the probability assigned to $w_j$ by the pre-trained model; we also rank the candidate replacements $w_j$ of each $w_i$ by $p(w_j)$. We set $P = 0.95$, based on preliminary experiments on development data. Table 6.8 shows that BIOLM-based substitution is almost as good as WORD2VEC-based substitution (94.45 vs. 95.60), but for BIOLM, the best performance is obtained with 50k artificial examples (compared to 10k for WORD2VEC). This is probably due to the fact that BIOLM suggests words that fit the particular context of the word being replaced and may, thus, suggest words with very different meanings that can be used in the particular context, adding noisy examples. By contrast, when using WORD2VEC we compare more directly each original word with candidate replacements.

| **DA with word substitution based on BIOLM** | |
|---|---|
| ID | Instance |
| 27789693 | **BIOASQ question:** Which database associates human noncoding SNPs with their three-dimensional interacting genes?<br>**BIOASQ sbippet:** 3DSNP: a database for linking human noncoding SNPs to their three-dimensional interacting genes.<br>**BIOASQ snippet after BIOLM substitution:** 3DSNP: a method for linking functional GWAS SNPs to their three-dimensional structural structures<br>**BIOASQ answer:** 3DSNP |

Table 6.7 Training instance generated via word substitution based on BIOLM. We randomly select at most 10 words of a BIOASQ snippet and we substitute each word $w_i$ with the most probable word $w_j$ suggested by BIOLM after masking $w_i$. Highlights of the same color indicate substituted words and the corresponding substitutions.

### Context

In the original training question-snippet-answer $\langle q, s, a \rangle$ triples, $s$ is usually a single sentence. To help the QA model learn to better distinguish relevant from irrelevant parts of the given snippet, we experimented with an additional DA method, where we find the original article that $s$ comes from and we expand $s$ with the $k_1$ (and $k_2$) sentences preceding (and following) it.[6] For each original $\langle q, s, a \rangle$ triple, we create multiple new $\langle q, s', a \rangle$ artificial triples, for

[5] We did not use BIOLM as an off-the-shelf QA model (Section 6.3.2), because it was not available fine-tuned on SQUAD.

[6] In BIOASQ, each gold snippet is accompanied by the PUBMED id of the article it was extracted from.

| ALBERT (SQUAD-V2) | +train ex. | PRAUC (BIOASQ dev) |
|---|---|---|
| +BIOASQ | 2,848 | 89.57 |
| +BIOLM | 2,848 | 91.76 |
| +BIOLM +BIOASQ | 5,696 | 92.37 |
| +BIOLM | 10,000 | 94.06 |
| +BIOLM +BIOASQ | 12,848 | 94.06 |
| +BIOLM | 30,000 | 93.63 |
| +BIOLM +BIOASQ | 32,848 | 93.75 |
| +BIOLM | 50,000 | 93.94 |
| +BIOLM +BIOASQ | 52,848 | **94.45** |
| +BIOLM | 100,000 | 93.79 |
| +BIOLM +BIOASQ | 102,848 | 93.84 |

Table 6.8 Data augmentation with *word substitution* based on *masked language modeling* using BIOLM.

different values of $k_1 \geq 0$ and $k_2 \geq 0$, such that $k_1 + k_2 = K$.[7] We experiment with $K = 2$ (three new triples for each original one); then to obtain more artificial examples, we repeat with $K = 4$ (five new triples for each original). To avoid truncation of the input examples, we remove all artificial examples that exceed 500 characters in length. For $K \in \{2, 4\}$, we obtain a development PRAUC score of 94.21 (Table 6.10), which is surpassed only by the two embedding-based word substitution methods (Tables 6.6–6.8). This DA method was introduced by Yoon et al. [213] who used it in BIOASQ.[8]

Adding surrounding text to the original data improves the results by a wide margin from 89.57 to 94.21 (Table 6.10). Even using just 2 surrounding sentences improves the results by 4 points. The best approach however is to increase the text by adding both 2 and 4 surrounding sentences, achieving a dev PRAUC of 94.21 even if the original BIOASQ data that contain only single sentences are not used (94.20 dev PRAUC).

### Back Translation

Back translation (BTR) has been used for data augmentation in several NLP tasks [53, 172]. The training examples are machine-translated from a source to a pivot language and back, obtaining paraphrases. We initially used French as the pivot language, then also Spanish and German, always translating from English to a pivot language and back with Google Translate. For each question-snippet-answer training triple of BIOASQ, we generate two new triples by

[7] Simply setting $k_1 = k_2$ would risk misguiding the model to always prefer the central sentence. We also experimented with adding *random* $k_1$ (or $k_2$) sentences before (and after) $s$, but the performance was much worse, possibly because the random sentences led to inferior context-aware token embeddings.

[8] Yoon et al. [213] reported an improvement in BIOASQ's Lenient Accuracy by 2.49 percentage points.

| **DA by adding context** | |
|---|---|
| ID | Instance |
| 15149039 | **BIOASQ question:** Which metabolite activates AtxA?<br>**BIOASQ snippet:** Transcription of the major Bacillus anthracis virulence genes is triggered by CO2, a signal mimicking the host environment.<br>**BIOASQ snippet with additional context:** Transcription of the major Bacillus anthracis virulence genes is triggered by CO2, a signal mimicking the host environment. A 182-kb plasmid, pXO1, carries the anthrax toxin genes and the genes responsible for their regulation of transcription, namely atxA and, pagR, the second gene of the pag operon. AtxA has major effects on the physiology of B. anthracis. It coordinates the transcription activation of the toxin genes with that of the capsule biosynthetic enzyme operon, located on the second virulence plasmid, pXO2. In rich medium, B. anthracis synthesises alternatively two S-layer proteins (Sap and EA1).<br>**Answer:** CO2 |

Table 6.9 A training instance generated by adding context around the original BIOASQ snippet. In the generated snippet the original one is highlighted.

| **ALBERT (SQUAD-V2)** | **+train ex.** | **PRAUC (BIOASQ dev)** |
|---|---|---|
| +BIOASQ | 2,848 | 89.57 |
| +CONTEXT ($K = 2$) | 4,568 | 93.91 |
| +CONTEXT ($K = 2$) +BIOASQ | 7,416 | 94.05 |
| +CONTEXT ($K \in \{2,4\}$) | 6,428 | 94.20 |
| +CONTEXT ($K \in \{2,4\}$) +BIOASQ | 9,276 | **94.21** |

Table 6.10 Data augmentation by *adding context to the snippet* ($K = 2$ or $K \in \{2,4\}$ surrounding sentences).

back-translating either the question or the snippet. If a new triple is identical to the original one, we discard it. We obtained 4,901 new training examples pivoting only to French, and 15,593 when also pivoting to Spanish and German.

Table 6.12 shows the results of data augmentation using back translation. When we use only French as our target language and create the same number of training examples as the original BIOASQ dataset and augment the training corpus we achieve the best results (92.95 PRAUC) for the development set. Competitive results can be achieved using all three target languages (French, Spanish, and German) even when we do not use the original corpus but only the translated data (92.21 PRAUC). However, it is observed that translated data in most cases do not have the same quality as the original data, since adding the original data in the training corpus always improves the results (92.95 from 91.84 PRAUC, 91.44 from 89.80 PRAUC and 89.99 from 89.80 PRAUC).

| DA via snippet back-translation | |
|---|---|
| ID | Instance |
| 8699317 | **Pivot language:** French<br>**BIOASQ question:** Which is the protein implicated in Spinocerebellar ataxia type 3?<br>**BIOASQ snippet:** Ataxin-3 (AT3) is the protein that triggers the inherited neurodegenerative disorder spinocerebellar ataxia type 3 when its polyglutamine (polyQ) stretch close to the C-terminus exceeds a critical length<br>**Back-translated snippet:** Ataxin-3 (AT3) is the protein that triggers spinocerebellar ataxia type 3 in inherited neurodegenerative disorder when its polyglutamine (polyQ) stretches near the C-terminus exceeds a critical length.<br>**BIOASQ answer:** Ataxin-3 |

Table 6.11 A training instance generated via back-translation of BIOASQ snippets using French as a pivot language. The generated instance contains a back-translated snippet and the corresponding BIOASQ question and answer.

| ALBERT (SQUAD-V2) | +train ex. | PRAUC (BIOASQ dev) |
|---|---|---|
| +BIOASQ | 2,848 | 89.57 |
| +BTR [FR ] | 2,848 | 91.84 |
| +BTR [FR ] +BIOASQ | 5,696 | **92.95** |
| +BTR [FR ] | 4,901 | 89.80 |
| +BTR [FR ] +BIOASQ | 7,749 | 91.44 |
| +BTR [FR,ES,DE ] | 2,848 | 89.80 |
| +BTR [FR,ES,DE ] +BIOASQ | 5,696 | 89.99 |
| +BTR [FR,ES,DE ] | 14,229 | 92.21 |
| +BTR [FR,ES,DE ] +BIOASQ | 17,077 | 92.21 |

Table 6.12 Data augmentation via *back-translation* (BTR), using one (FR) or three (FR, ES, DE) pivot languages.

### Question Generation

Question generation (QG) has been found an effective DA method in open-domain MRC [8, 28, 108]. The main reported benefit is that it increases the diversity of questions [151, 179]. Typically QG models are fed with a snippet $s$, select an answer span $a$ of $s$, and generate a question $q$ answered by $a$. We take the T5 [152] encoder-decoder Transformer model fine-tuned for QG on a modified version of SQUAD by Enrico et al. [108][9] and use it to generate alternative questions $q'$ and answer spans $a'$ from the snippets $s$ of the BIOASQ $\langle q, s, a \rangle$ training triples, producing artificial $\langle q', s, a' \rangle$ triples. Multiple artificial triples can be

[9]The T5 QG model we used is available at `https://github.com/patil-suraj/question_generation`.

generated from the same original one (the same $s$), but we require each $q'$ to be answered by a different answer span $a'$ to maximize the diversity of questions. We obtained 3,389 artificial triples from the 2,848 original ones this way. An alternative we explored is to select random snippets $s$ from random PUBMED abstracts, and use the QG model to produce artificial $\langle q', s, a' \rangle$ triples. The alternative approach can generate millions of artificial triples; we generated up to 100k.

| **DA via Question Generation using BIOASQ snippets** | |
|---|---|
| ID | Instance |
| 16800744 | **Generated question:** What is the human tissue kallikrein family of?<br>**BIOASQ snippet:** The human tissue kallikrein family of serine proteases (hK1-hK15 encoded by the genes KLK1-KLK15) is involved in several cancer-related processes.<br>**Generated answer:** serine proteases |

Table 6.13 A training instance generated using T5. Given a BIOASQ snippet T5 selects a span of the snippet and generates a question that can be answered by that span. We select spans different than the ones used in BIOASQ.

| **DA via Question Generation using random snippets from random PUBMED abstracts** | |
|---|---|
| ID | Instance |
| 30706485 | **Generated question:** What were connected to a volume-cycled ventilator after sedation, analgesia and endotracheal intubation?<br>**PUBMED snippet:** After sedation, analgesia and endotracheal intubation, pigs were connected to a volume-cycled ventilator.<br>**Generated answer:** pigs |

Table 6.14 A training instance generated using T5. Given a random snippet from a random PUBMED article, T5 selects a span of the snippet and generates a question that can be answered by that span.

Using the T5 model to automatically create training examples slightly improves the results from 89.57 when using only the original dataset to 90.69 PRAUC for the development set (Table 6.15) when using additional $50k$ training examples. Using only data created from the T5 model does not surpass the results of the model trained only on the original data. This is expected as the questions found in the augmented data may not resemble the original questions of the BIOASQ dataset and may be too simplistic.

| ALBERT (SQUAD-V2) | +train ex. | PRAUC (BIOASQ dev) |
|---|---|---|
| +BIOASQ | 2,848 | 89.57 |
| +T5 @BIOASQ | 3,389 | 84.46 |
| +T5 @BIOASQ +BIOASQ | 6,237 | 88.46 |
| +T5 @PUBMED | 2,848 | 85.79 |
| +T5 @PUBMED +BIOASQ | 5,696 | 89.29 |
| +T5 @PUBMED | 10,000 | 87.30 |
| +T5 @PUBMED +BIOASQ | 12,848 | 89.34 |
| +T5 @PUBMED | 30,000 | 86.65 |
| +T5 @PUBMED +BIOASQ | 32,848 | 90.51 |
| +T5 @PUBMED | 50,000 | 87.30 |
| +T5 @PUBMED +BIOASQ | 52,848 | **90.69** |
| +T5 @PUBMED | 100,000 | 87.30 |
| +T5 @PUBMED +BIOASQ | 102,848 | 90.61 |

Table 6.15 Data augmentation via *question generation* using T5. Questions are generated from the training snippets of BIOASQ (T5 @BIOASQ) or from random snippets from random PUBMED abstracts (T5 @PUBMED).

### Information Retrieval

Data augmentation based on Information Retrieval (IR) has been found promising in previous open-domain QA work [205].[10] Given a question and a gold answer, the question is used as a query to an IR system. Any retrieved document (or passage therein) that includes the gold answer is used to construct a new training example (with the same question and gold answer). We used the open data from the PUBMED Baseline Repository[11] to create a pool of 22.3M biomedical documents. Each document is the concatenation of the title and abstract of a PUBMED article. We indexed all documents with an ElasticSearch retrieval engine[12] and used the 500 top ranked (by BM25) documents per question. From the original 2,848 question-snippet-answer triples, only 289 more were generated, because in most of the retrieved documents no sentence included the entire answer (individual terms of the answer might be scattered in the document). We suspect that the biomedical experts of BIOASQ create questions whose answers cannot be found in large numbers of documents (unlike common questions in open-domain QA datasets), and the few relevant documents (and snippets) of each question have already been included in the BIOASQ training data.

[10] [205] gained 2.7 to 9.7 F1 percentage points (pp.) in all four datasets they experimented with.

[11] `lhncbc.nlm.nih.gov/ii/information/MBR.html`

[12] `https://www.elastic.co/`

Table 6.17 shows that IR-based augmentation led to very minor gains in our case, because of the very few additional instances.

| **DA via Information Retrieval** | |
|---|---|
| ID | Instance |
| 25941473 | **BIOASQ question:** Which is the neurodevelopmental disorder associated to mutations in the X- linked gene mecp2?<br>**Retrieved snippet:** Genotype-specific effects of Mecp2 loss-of-function on morphology of Layer V pyramidal neurons in heterozygous female Rett syndrome model mice.<br>**BIOASQ answer:** rett syndrome |

Table 6.16 A training instance generated via IR. A BIOASQ question is used as the query to retrieve PUBMED documents. For each snippet of the retrieved documents that contains the answer, we generate a new training triplet consisting of the BIOASQ question, the snippet, and the BIOASQ answer.

Data augmentation using IR improves results from 89.57 to 89.80 PRAUC for the development set when combined with the original BIOASQ data. Unfortunately in the entire PUBMED database, only 289 examples could be created from the retrieved documents when the question is used as a query.

| **ALBERT (SQUAD-V2)** | **+train ex.** | **PRAUC (BIOASQ dev)** |
|---|---|---|
| +BIOASQ | 2,848 | 89.57 |
| +IR | 289 | 80.30 |
| +IR +BIOASQ | 3,137 | **89.80** |

Table 6.17 Data augmentation via *information retrieval* (IR), using PUBMED titles and abstracts as documents.

## 6.4 Results & Analysis

### 6.4.1 BIOASQ Results

In Table 6.18 we present the precision-recall AUC for both development and test set. For reasons of space economy, we only report for each DA method, the results of the model that obtained the best development performance, among models obtained using different numbers of artificial training instances. Overall we observe that all DA strategies improve the results for both development and test set. Test scores are worse than the development

scores which is expected as early stopping is applied and the model from the epoch with the best development score is selected. However, the improvement (shown in parenthesis) of the scores when using DA is higher in the test set.

| Method | +train ex. | PRAUC (dev) | PRAUC (test) |
|---|---|---|---|
| ALBERT (SQUAD-V2) | 0 | 80.25 | 77.78 |
| + BIOASQ | 2,848 | 89.57 | 76.78 |
| +WORD2VEC +BIOASQ | 12,848 | **95.60 (+6.03)** | **84.99 (+8.21)** |
| +BIOLM +BIOASQ | 52,848 | 94.45 (+4.88) | 82.76 (+5.98) |
| +CONTEXT +BIOASQ | 9,276 | 94.21 (+4.64) | 81.63 (+4.85) |
| +BIOMRC +BIOASQ | 12,848 | 93.15 (+3.58) | 82.04 (+5.26) |
| +BTR +BIOASQ | 18,441 | 92.66 (+3.09) | 81.27 (+4.49) |
| +T5 @PUBMED +BIOASQ | 52,848 | 90.69 (+1.12) | 80.26 (+3.48) |
| +IR +BIOASQ | 3,137 | 89.80 (+0.23) | 78.66 (+1.88) |

Table 6.18 Performance of DA methods on *development* and *test* data, ordered by decreasing development score. For each DA method, we use the configuration (from Tables 6.4–6.10) with the best development score.

We employed ten-fold stratified cross-validation to evaluate the results based on micro-PRAUC. Ten-fold stratified cross-validation is a model evaluation method used in machine learning to assess the performance of a model. The dataset is divided into ten equal parts ('folds') and the model is trained and evaluated on each fold, one at a time. The process is repeated ten times, with each fold serving as the test set once. The advantage of this method is that all samples are used for both training and testing, ensuring that all parts of the dataset are used and that the model has seen all the examples at least once. In comparison to the previous evaluation method, ten-fold cross-validation uses ten folds instead of just one. However, this leads to a ten-fold increase in the amount of time needed to complete a single experiment.

We evaluate all DA strategies except IR, which produced the smallest number of additional training examples and contributed less in the results of Table 6.18. We also use always $10k$ additional training examples in every DA experiment, to achieve a fair comparison for context increasing and back translation techniques which produce fewer training examples. After ten-fold cross-validation we observe once more that all DA strategies improve the results.

Surprisingly, increasing the length of the processed snippet by adding surrounding sentences improves the results and surpasses all data augmentation techniques. As seen in Table 6.20 there are some cases where the snippet is short in length, and even though

| Method | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|
| No AUG | 74.62 | 34.82 | 77.19 | 35.08 |
| CONTEXT | **79.27 (+4.65)** | 60.85 | **81.17 (+3.98)** | 63.86 |
| WORD2VEC | 79.03 (+4.41) | 60.68 | 80.41 (+3.22) | 63.94 |
| BIOLM | 78.71 (+4.09) | **62.82** | 80.52 (+3.33) | **68.00** |
| BTR | 77.69 (+3.07) | 62.34 | 79.99 (+2.80) | 66.92 |
| BIOMRC | 76.97 (+2.35) | 55.24 | 79.85 (+2.66) | 58.33 |
| T5 | 76.50 (+1.88) | 56.60 | 79.27 (+2.08) | 59.82 |

Table 6.19 10-fold cross validation results for BIOASQ.

it contains the answer there is not much context. However, by increasing the context (adding two more sentences) we observe that there is a lot more context which also shares multiple tokens with the question so the QA model performs better using context-based data augmentation. Small sentences without enough context are not present in BIOASQ data since the biomedical experts were asked to annotate snippets that contain enough information to answer the questions. This might explain why WORD2VEC data augmentation performed better in BIOASQ than context-based data augmentation.

Substituting tokens using language models also improves the results and constitutes the best data augmentation technique when adding context cannot be applied as an augmentation technique. Using pre-trained WORD2VEC embeddings for data augmentation also competes with data augmentation using the BIOLM language model despite their training method being more simplistic. BIOMRC and T5 question generation were ranked last as data augmentation techniques even though they both improve model results. However, in all experiments, we used only $10k$ new training examples even if we could produce seemingly unlimited new artificial training data.

### 6.4.2 COVIDQA Results

Results on BIOASQ were very encouraging for biomedical factoid QA. To ensure that our previous results were not BIOASQ-specific, we decided to experiment with another biomedical QA dataset. Similarly to BIOASQ, COVIDQA [121] includes biomedical questions annotated by volunteer biomedical experts on scientific articles. However, COVIDQA uses only articles related to COVID-19. The experts examined 147 biomedical articles and created multiple question/answer pairs for each article. In total $2,019$ question/answer pairs were created.

COVIDQA includes multiple answers that do not fit our factoid QA setting as the answers are too long. We discarded all answers that exceeded 40 characters and answers that included only one character. For each paragraph, multiple occurrences of the answer could be found.

| An instance from the COVIDQA dataset | |
|---|---|
| ID | Instance |
| 83 | **Question:** What potential mechanism, could be presumed to underlie the pathogenesis of HCPS?.<br>**Answer:** Innate immune mechanisms<br>**Snippet:** These include: (1) Innate immune mechanisms.<br>**Snippet Augmented with WORD2VEC:** These involve: (1) Innate immune mechanisms.<br>**Snippet with context:** It is tempting to speculate that mediators produced in the lung in connection with the inflammatory infiltrate can percolate through the coronary circulation with minimal dilution in HCPS, a disadvantageous consequence of the close anatomic juxtaposition of the two organs. Thus, at least three classes of potential mechanisms, some overlapping and all certainly nonexclusive of the others, could be presumed to underlie the pathogenesis of HCPS. These include: (1) Innate immune mechanisms. |

Table 6.20 An instance of the COVIDQA dataset along with an instance created using WORD2VEC data augmentation and context-based data augmentation. We observe that when the context is increased there are more tokens shared between the augmented snippet and the question.

We applied sentence splitting and created multiple examples that include a question a snippet, and an answer so that the resulting instances from the COVIDQA dataset resemble the format of the instances we used in BIOASQ experiments. Using the same format allows us also to evaluate on COVIDQA data, the already fine-tuned models on BIOASQ data (see also BEST-BIOASQ in Table 6.21) which we further discuss below. According to these criteria we ended up using $2,024$ examples.

| Method | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|
| No AUG | 58.47 | 6.47 | 55.11 | 6.82 |
| BIOLM | **66.79** | **30.64** | 59.65 | 25.35 |
| WORD2VEC | 65.17 | 22.29 | **61.74** | 20.56 |
| CONTEXT | 64.70 | 27.97 | 60.78 | **27.15** |
| BIOMRC | 61.69 | 17.14 | 55.52 | 15.38 |
| BEST-BIOASQ | 67.97 | **46.14** | **67.57** | **45.60** |
| BEST-BIOASQ + BIOLM | **68.38** | 40.01 | 65.67 | 37.87 |
| BEST-BIOASQ + WORD2VEC | 66.27 | 38.55 | 61.97 | 35.38 |

Table 6.21 10-fold cross validation results for COVIDQA.

We once again start with the ALBERT model pre-trained only on SQUAD data. We applied our data augmentation techniques to the COVIDQA data and applied 10-fold cross-validation similarly to BIOASQ experiments to evaluate model performance. In Table 6.21 we observe

that once again all data augmentation techniques improve the results. On average BIOLM achieved the best PRAUC score in the development set (0.6679) and WORD2VEC substitution achieved the best test PRAUC score (0.6174) confirming that word substitution is a strong data augmentation technique for biomedical factoid QA.

Instead of using the ALBERT model, which was pre-trained only on SQUAD data, we also experimented using the model that performed best across all folds on the development set of BIOASQ data. In the bottom part of Table 6.21 we observe that upon further fine-tuning the best BIOASQ model with COVIDQA data the results further improve compared to the results of the models that were only trained with COVIDQA data (from 65.17 to 66.27 for WORD2VEC and from 66.79 to 68.38 for the BIOLM approach). This is expected as the model has been exposed to more training examples for the biomedical factoid QA task. It is however strange that the BEST-BIOASQ model that was trained only on COVIDQA data performed better than the BEST-BIOASQ model that was trained on both the original COVIDQA and the augmented data (67.57 TEST PRAUC when using the original COVIDQA data, 65.67 TEST PRAUC when the original COVIDQA data and BIOLM data augmentation and 61.97 when the original COVIDQA data and WORD2VEC data augmentation). It might be the case that since COVIDQA was constructed using only 147 documents the BEST-BIOASQ model finally overfits on training data and therefore performs poorly on test data.

## 6.5 Conclusions and Future work

In this chapter, we presented a novel factoid question-answering model that can extract multiple text spans as answers to a given question. This is a significant contribution to the field of biomedical NLP as it is the first time that seven data augmentation techniques have been compared in this domain. We evaluated our model on two biomedical datasets, BIOASQ and COVIDQA, and found that all data augmentation techniques improved the results in both datasets. Additionally, fine-tuning the model on BIOASQ and then on COVIDQA resulted in even further improvement in performance on the COVIDQA dataset. Overall, this work presents new insights and methodologies for improving the performance of factoid question-answering models in biomedical NLP.

The current work on data augmentation in NLP has a few limitations, which are also common in the field. Firstly, the focus is on data augmentation in the input space, meaning the artificially generated data is in text form, similar to the original data. Secondly, the data augmentation is performed only once, before training, as opposed to generating new artificial instances for each training epoch. These limitations, although common, lead to reduced computational costs and enable sharing of augmented datasets.

However, there is room for improvement in the current approach. Future work can include online data augmentation and feature space data augmentation [29, 43, 172] to expose the model to more synthetic data and act as layer-specific regularization. Additionally, the study of active learning [51, 115] can be incorporated to select the most informative, diverse, and representative artificial training instances generated by data augmentation. The effect of data augmentation in few-shot learning can also be studied using small subsets of the BIOASQ dataset.

# Chapter 7

# Conclusions and Future Work

## 7.1 Summary of the thesis and its contributions

Our research aimed to address three major research questions in the field of biomedical natural language processing (NLP). The first question centered around document and snippet retrieval in the biomedical domain. The goal was to develop methods for efficiently and accurately retrieving relevant information from large collections of biomedical documents. The second research question concerned the creation of big corpora for machine reading comprehension in the biomedical domain. Here, we aimed to build large datasets that can be used to train deep learning models for machine reading comprehension tasks. The third and final research question focused on data augmentation for biomedical factoid question answering. The goal here was to explore and compare different data augmentation techniques for factoid question answering in the biomedical domain.

Working in the biomedical NLP domain is particularly challenging due to the highly specialized and technical language used in biomedical texts. However, the potential benefits of advances in this field are substantial, as they can aid in the development of new medical treatments and therapies. Our research is therefore valuable in that it contributes to the ongoing efforts to make the vast amounts of biomedical information more accessible and understandable to both experts and non-experts. Our work on document and snippet retrieval can help healthcare practitioners quickly locate relevant information for patient care, while our work on machine reading comprehension and data augmentation for question answering can help in the development of intelligent systems that can assist with medical diagnosis and treatment.

In Chapter 4 of our research, we aimed to tackle the critical challenge of document and snippet retrieval in the biomedical domain. To address this challenge, we proposed a novel architecture that jointly ranks documents and snippets with respect to a given question.

This architecture can be utilized with any neural text relevance model and is designed to be highly flexible and adaptable to a wide range of applications. Our research results were highly successful, with our models demonstrating vast improvements over existing pipelines in terms of snippet retrieval performance while also remaining competitive in document retrieval. In addition, we provided a modified version of the Natural Questions dataset, modified for document and snippet retrieval thus creating a valuable resource for the community. Furthermore, our models were deployed in two real-world use cases to aid biomedical experts, such as answering questions related to the COVID pandemic and performing literature identification and screening for systematic reviews. Importantly, our models were recognized for their exceptional performance, winning prizes for document retrieval and snippet retrieval in BIOASQ-7 [143] (see also the results of the competition [127]), BIOASQ-8 [144] (see also the results of the competition [129]), and BIOASQ-9 (see also the results of the competition [128]), respectively.

In chapter 5 of our research, we made significant contributions to the field of Machine Reading Comprehension (MRC) in the biomedical domain. MRC is a subfield of Natural Language Processing (NLP) and is related to Question Answering (QA) in that it aims to understand and extract relevant information from a given document to answer questions. We constructed and made publicly available two large MRC datasets, namely BioRead and BIOMRC, which consist of 16.4 million and 800 thousand instances respectively. BioRead is one of the largest MRC datasets to date and the first of its kind in the biomedical domain. To demonstrate the effectiveness of our datasets, we re-implemented and tested two well-known MRC methods (AS-READER and AOA-READER) and compared them against four baselines, as a first step towards a biomedical MRC leaderboard. Additionally, we introduced two new deep learning models for MRC and showed that the best-performing model was able to surpass non-experts in selecting the correct answer to questions in a small-scale experiment using 60 examples from the BIOMRC dataset. Finally, we showed that BIOMRC could be used as a data augmentation method to improve results for Factoid Question Answering, highlighting the usefulness of the artificially created dataset.

Finally, in chapter 6 of our work, we studied the impact of Data Augmentation (DA) on biomedical Question Answering (QA) with a focus on the factoid questions in the BIOASQ challenge. We started by evaluating pre-trained models that were already fine-tuned on general domain QA data and selected the best model available. We modified the architecture of the chosen model (ALBERT) to enable the identification of multiple answer spans in the text. Furthermore, we performed additional fine-tuning on the BIOASQ dataset, in addition to its pre-existing fine-tuning on the generic question-answering (SQUAD) dataset, to attain a more robust baseline. Our experiments showed that the simplest DA method considered,

word substitution, was the best data augmentation method that improved both development and test performance significantly. We also verified the effectiveness of word substitution through experiments on the COVIDQA dataset. The performance of the model was further improved when it was fine-tuned on both the BIOASQ and COVIDQA datasets.

## 7.2 Future work

There is a large scope for future work in the field of Biomedical Question Answering (QA) using Natural Language Processing (NLP) techniques. In recent times, several deep learning models with millions of trainable parameters have been developed and trained in the biomedical domain, such as BioElectra[79], BioMegatron[170], GatorTron[207], MED-PALM [174], BioGPT [110] and BioBART [214]. An interesting study that could be conducted would be to replace the transformer-based models used in our research with these pre-trained biomedical language models and evaluate the performance of our models that use the embeddings produced by these models. Another improvement could be achieved by fine-tuning the transformer-based models that we used in our experiments to see if there is any improvement in their performance. We did not perform these experiments due to time and computational limitations.

Multiple questions of the BIOASQ dataset are answered either by the name of a gene or the name of the disease. Therefore external knowledge could be used to identify these biomedical entities in the processed texts, such as the International Classification of Diseases (ICD[1]), which includes names and synonyms of all human-related diseases or the Genecards ontology [161], which includes metadata about genomes of many species. In the examined texts, the use of these vocabularies enables the identification of diseases and genes and helps to narrow down the entities to a single one when a question specifically asks for that type of entity. Alternatively, a knowledge graph could be used, such as the extensive DRKG[70] or a smaller biomedical graph containing specific entities [82] to identify and tag more biomedical entities and identify relations between the entities of the question and the examined text. The utilization of knowledge graphs has the potential to provide answers to questions that inquire about the relationship between two entities, specifically in regards to a specific relation. For example, the question 'Can LB-100 downregulate miR-33?' taken from the BIOASQ dataset can be answered by a graph where the entity 'LB-100' is linked with the entity 'miR-33' by the relation 'downregulation' in DRKG.

In both retrieval models and factoid QA models an issue that could be addressed is the identification and replacement of the abbreviated forms of phrases in a biomedical text.

[1]The current version of the ontology (ICD11) is described in: `https://icd.who.int/en`.

For example in the snippets provided by BIOASQ there is a question that requires the text 'essential tremor' as an answer, however, in some snippets, only the abbreviated form ('ET') can be found[2]. Therefore even if the model selects the abbreviation as an answer it would be identified as an incorrect answer and the model would be penalized. The use of abbreviations in language models (like the pre-trained language models used in all of our experiments) can pose a challenge as they can represent multiple entities in a text. To address this issue, it is necessary to replace abbreviations with their full forms in order to ensure accurate embedding representation.

All proposed deep learning models for factoid Question Answering (QA) are limited to the knowledge present in the input snippet. The language models encode the tokens in their context and the contextual embeddings are then used to extract spans from the text as answers to a given question. The vast knowledge that a biomedical expert has obtained through years of training and the common sense that humans possess are mainly represented by the large language models. A possible solution to enhance the current state of factoid QA is to integrate the knowledge from biomedical graphs with biomedical language models, similar to work from Yasunaga et al. [211].

[2]You can visit PUBMED to read the corresponding abstract `https://pubmed.ncbi.nlm.nih.gov/29481820/`

# References


[1] Abdel-Nabi, H., Awajan, A., and Ali, M. Z. (2022). Deep learning-based question answering: a survey. *Knowledge and Information Systems*.

[2] Abdollahi, M., Gao, X., Mei, Y., Ghosh, S., and Li, J. (2020). Ontology-guided data augmentation for medical document classification. In *Artificial Intelligence in Medicine*, pages 78–88, Cham. Springer International Publishing.

[3] Adam, G. P., Pappas, D., Papageorgiou, H., Evangelou, E., and Trikalinos, T. A. (2022). A novel tool that allows interactive screening of PubMed citations showed promise for the semi-automation of identification of Biomedical Literature. *Journal of Clinical Epidemiology*, 150:63–71.

[4] Aggarwal, A., Tam, C. C., Wu, D., Li, X., and Qiao, S. (2022). Artificial Intelligence (AI)-based Chatbots in Promoting Health Behavioral Changes: A Systematic Review. *medRxiv*.

[5] Ahmad, A., Constant, N., Yang, Y., and Cer, D. (2019). ReQA: An evaluation for end-to-end answer retrieval models. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 137–146, Hong Kong, China.

[6] Ai, Q., O'Connor, B. T., and Croft, W. B. (2018). A Neural Passage Model for Ad-hoc Document Retrieval. In *Advances in Information Retrieval*.

[7] Alamri, A. and Stevenson, M. (2016). A corpus of potentially contradictory research claims from cardiovascular research abstracts. In *Journal of Biomedical Semantics*.

[8] Alberti, C., Andor, D., Pitler, E., Devlin, J., and Collins, M. (2019). Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

[9] Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., Tepper, N., and Zwerdling, N. (2020). Do not have enough data? deep learning to the rescue! *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7383–7390.

[10] Aronson, A. R. and Lang, F.-M. (2010). An overview of MetaMap: historical perspective and recent advances. *JAMIA*, 17:229–236.

[11] Bajgar, O., Kadlec, R., and Kleindienst, J. (2016). Embracing data abundance: BookTest Dataset for Reading Comprehension. *CoRR*, abs/1610.00956.

[12] Bardhan, J., Colas, A., Roberts, K., and Wang, D. Z. (2022). Drugehrqa: A question answering dataset on structured and unstructured electronic health records for medicine related queries. *PhysioNet*.

[13] Bayer, M., Kaufhold, M., and Reuter, C. (2021a). A survey on data augmentation for text classification. *CoRR*, abs/2107.03158.

[14] Bayer, M., Kaufhold, M.-A., Buchhold, B., Keller, M., Dallmeyer, J., and Reuter, C. (2021b). Data augmentation in natural language processing: A novel text generation approach for long and short text classifiers. *ArXiv*, abs/2103.14453.

[15] Belinkov, Y. and Bisk, Y. (2018). Synthetic and natural noise both break neural machine translation. *ArXiv*, abs/1711.02173.

[16] Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: Pretrained Language Model for Scientific Text. In *EMNLP*.

[17] Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

[18] Ben Abacha, A. and Demner-Fushman, D. (2019). A question-entailment approach to question answering. *BMC Bioinform.*, 20(1):511.

[19] Ben Abacha, A., Shivade, C., and Demner-Fushman, D. (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*, pages 370–379, Florence, Italy.

[20] Bird, S., Edward, L., and Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

[21] Boytsov, L., Novak, D., Malkov, Y., and Nyberg, E. (2016). Off the beaten path: Let's replace term-based retrieval with k-nn search. *Computing Research Repository (CoRR)*, abs/1610.10001.

[22] Brabra, H., Báez, M., Benatallah, B., Gaaloul, W., Bouguelia, S., and Zamanirad, S. (2022). Dialogue management in conversational systems: A review of approaches, challenges, and opportunities. *IEEE Transactions on Cognitive and Developmental Systems*, 14(3):783–798.

[23] Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159.

[24] Brokos, G., Liosis, P., McDonald, R., Pappas, D., and Androutsopoulos, I. (2018). AUEB at BioASQ 6: Document and Snippet Retrieval. In *Proceedings of the 6th BioASQ Workshop*, pages 30–39, Brussels, Belgium.

[25] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *ArXiv*, abs/2005.14165.

[26] Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

[27] Cai, H., Chen, H., Song, Y., Zhang, C., Zhao, X., and Yin, D. (2020). Data manipulation: Towards effective instance learning for neural dialogue generation via learning to augment and reweight. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6334–6343, Online. Association for Computational Linguistics.

[28] Chan, Y.-H. and Fan, Y.-C. (2019). A recurrent BERT-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162, Hong Kong, China. Association for Computational Linguistics.

[29] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321—-357.

[30] Chen, D., Bolton, J., and Manning, C. D. (2016). A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany.

[31] Chen, D., Fisch, A., Weston, J., and Bordes, A. (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)*, pages 1870–1879, Vancouver, Canada.

[32] Chen, J., Shen, D., Chen, W., and Yang, D. (2021). Hiddencut: Simple data augmentation for natural language understanding with better generalization. *ArXiv*, abs/2106.00149.

[33] Cheng, Y., Jiang, L., Macherey, W., and Eisenstein, J. (2020). Advaug: Robust adversarial augmentation for neural machine translation. *ArXiv*, abs/2006.11834.

[34] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

[35] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

[36] Cui, L., Huang, S., Wei, F., Tan, C., Duan, C., and Zhou, M. (2017). SuperAgent: A customer service chatbot for E-commerce websites. In *Proceedings of ACL 2017, System Demonstrations*, pages 97–102, Vancouver, Canada. Association for Computational Linguistics.

[37] Cui, Y., Chen, Z., Wei, S., Wang, S., Liu, T., and Hu, G. (2016). Attention-over-Attention Neural Networks for Reading Comprehension. *CoRR*, abs/1607.04423.

[38] Dai, Y., Yu, H., Jiang, Y., Tang, C., Li, Y., and Sun, J. (2020). A survey on dialog management: Recent advances and challenges. *CoRR*, abs/2005.02233.

[39] Dai, Z., Xiong, C., Callan, J., and Liu, Z. (2018). Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 126–134, Marina Del Rey, CA.

[40] Danesh, S., Sumner, T., and Martin, J. H. (2015). SGRank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*.

[41] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[42] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[43] DeVries, T. and Taylor, G. W. (2017). Dataset augmentation in feature space.

[44] Dhingra, B., Danish, D., and Rajagopal, D. (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)*, pages 582–587, New Orleans, Louisiana.

[45] Dinçer, B. T. (2012). Irra at trec 2012: Divergence from independence (dfi). In *TREC*.

[46] Djenouri, Y., Belhadi, A., Djenouri, D., and Lin, J. C.-W. (2021). Cluster-based information retrieval using pattern mining. *Applied Intelligence*, 51(4):1888–1903.

[47] Dong, L., Mallinson, J., Reddy, S., and Lapata, M. (2017). Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886, Copenhagen, Denmark. Association for Computational Linguistics.

[48] Dror, R., Baumer, G., Shlomov, S., and Reichart, R. (2018). The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. In *Proceedings of the 56th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 1383–1392.

[49] Du, Y., Guo, W., and Zhao, Y. (2019). Hierarchical question-aware context learning with augmented data for biomedical question answering. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 370–375.

[50] Dunn, M., Sagun, L., Higgins, M., Güney, V. U., Cirik, V., and Cho, K. (2017). SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. *ArXiv*, abs/1704.05179.

[51] Ein-Dor, L., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., and Slonim, N. (2020). Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online.

[52] Fan, Y., Guo, J., Lan, Y., Xu, J., Zhai, C., and Cheng, X. (2018). Modeling Diverse Relevance Patterns in Ad-Hoc Retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

[53] Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., and Hovy, E. (2021). A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online.

[54] Goldberg, Y. (2017). *Neural Network Methods in Natural Language Processing*. Morgan and Claypool Publishers.

[55] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM.

[56] Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., Naumann, T., Gao, J., and Poon, H. (2021). Domain-specific language model pretraining for biomedical natural language processing. *ACM Trans. Comput. Healthcare*, 3(1).

[57] Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64, Indianapolis, Indiana, USA.

[58] Hannan, D., Jain, A., and Bansal, M. (2020). Manymodalqa: Modality disambiguation and QA over diverse inputs. *CoRR*, abs/2001.08034.

[59] Haridas, H. T., Fouda, M. M., Fadlullah, Z. M., Mahmoud, M., ElHalawany, B. M., and Guizani, M. (2022). MED-GPVS: A Deep Learning-Based Joint Biomedical Image Classification and Visual Question Answering System for Precision e-Health. In *ICC 2022 - IEEE International Conference on Communications*, pages 3838–3843, Seoul, Korea, Republic of. IEEE.

[60] Harter, S. P. (1975). A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science*, 26(4):197–206.

[61] Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching Machines to Read and Comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, page 1693–1701, Cambridge, MA, USA.

[62] Hill, F., Bordes, A., Chopra, S., and Weston, J. (2016). The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *CoRR*.

[63] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[64] Hosein, S., Andor, D., and McDonald, R. (2019). Measuring domain portability and error propagation in biomedical QA. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 686–694, Wurzburg, Germany.

[65] Huang, C.-C. and Lu, Z. (2016). Community challenges in biomedical text mining over 10 years: success, failure and the future. *Briefings in Bioinformatics*, 17(1):132–144.

[66] Huang, M.-S., Lai, P.-T., Tsai, R. T.-H., and Hsu, W.-L. (2020). Revised JNLPBA Corpus: A Revised Version of Biomedical NER Corpus for Relation Extraction Task. *Briefings in Bioinformatics*, 21:2219–2238. arXiv:1901.10219 [cs].

[67] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. P. (2013). Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

[68] Hui, K., Yates, A., Berberich, K., and de Melo, G. (2017). PACRR: A Position-Aware Neural IR Model for Relevance Matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark.

[69] Hui, K., Yates, A., Berberich, K., and de Melo, G. (2018). Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 279–287, Marina Del Rey, CA.

[70] Ioannidis, V. N., Song, X., Manchanda, S., Li, M., Pan, X., Zheng, D., Ning, X., Zeng, X., and Karypis, G. (2020). Drkg - drug repurposing knowledge graph for covid-19. `https://github.com/gnn4dr/DRKG/`.

[71] Islam, M. S., Kamal, A. H. M., Kabir, A., Southern, D. L., Khan, S. H., Hasan, S. M. M., Sarkar, T., Sharmin, S., Das, S., Roy, T., Harun, M. G. D., Chughtai, A. A., Homaira, N., and Seale, H. (2021). Covid-19 vaccine rumors and conspiracy theories: The need for cognitive inoculation against misinformation to improve vaccine adherence. *PLoS ONE*, 16.

[72] Jiang, Z., Chi, C., and Zhan, Y. (2021). Research on medical question answering system based on knowledge graph. *IEEE Access*, 9:21094–21101.

[73] Jin, D., Pan, E., Oufattole, N., Weng, W., Fang, H., and Szolovits, P. (2020). What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *CoRR*, abs/2009.13081.

[74] Jin, Q., Dhingra, B., Liu, Z., Cohen, W., and Lu, X. (2019). PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

[75] Jun, C., Jang, H., Sim, M., Kim, H., Choi, J., Min, K., and Bae, K. (2022). ANNA: Enhanced language representation for question answering. In *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 121–132, Dublin, Ireland. Association for Computational Linguistics.

[76] Jungiewicz, M. and Smywinski-Pohl, A. (2019). Towards textual data augmentation for neural networks: synonyms and maximum loss. *Computer Science*, 20.

[77] Kadlec, R., Schmid, M., Bajgar, O., and Kleindienst, J. (2016a). Text Understanding with the Attention Sum Reader Network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918, Berlin, Germany.

[78] Kadlec, R., Schmid, M., Bajgar, O., and Kleindienst, J. (2016b). Text Understanding with the Attention Sum Reader Network. *CoRR*, abs/1603.01547.

[79] Kanakarajan, K. r., Kundumani, B., and Sankarasubbu, M. (2021). BioELECTRA:pretrained biomedical text encoder using discriminators. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 143–154, Online. Association for Computational Linguistics.

[80] Khare, Y., Bagal, V., Mathew, M., Devi, A., Priyakumar, U. D., and Jawahar, C. (2021). MMBERT: Multimodal BERT Pretraining for Improved Medical VQA. *ArXiv*.

[81] Khattab, O. and Zaharia, M. (2020). ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT.

[82] Kotitsas, S., Pappas, D., Androutsopoulos, I., McDonald, R., and Apidianaki, M. (2019). Embedding biomedical ontologies by jointly encoding network structure and textual node descriptors. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 298–308, Florence, Italy. Association for Computational Linguistics.

[83] Kovaleva, O., Shivade, C., Kashyap, S., Kanjaria, K., Wu, J., Ballah, D., Coy, A., Karargyris, A., Guo, Y., Beymer, D. B., Rumshisky, A., and Mukherjee, V. M. (2020). Towards Visual Dialog for Radiology. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 60–69, Online. Association for Computational Linguistics.

[84] Krovetz, R. (1993). Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202, Pittsburgh, PA.

[85] Kumar, V., Choudhary, A., and Cho, E. (2020). Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

[86] Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. (2015). From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 957–966. JMLR.org.

[87] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural Questions: a Benchmark for Question Answering Research. *Transactions of the Association of Computational Linguistics*.

[88] Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. (2017). RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark.

[89] Lai, T. M., Bui, T., and Li, S. (2018). A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2132–2144, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

[90] Lan, Y., He, G., Jiang, J., Jiang, J., Zhao, W. X., and Wen, J. (2021). A survey on complex knowledge base question answering: Methods, challenges and solutions. *CoRR*, abs/2105.11644.

[91] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

[92] Leaman, R., Islamaj Doğan, R., and Lu, Z. (2013). DNorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917.

[93] LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[94] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2019a). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

[95] Lee, K., Chang, M.-W., and Toutanova, K. (2019b). Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy.

[96] Lehnert, W. G. (1980). The process of question answering - a computer simulation of cognition. *American Journal of Computational Linguistics*, 6(3-4).

[97] Lewis, M. and Fan, A. (2019). Generative question answering: Learning to answer the whole question. In *International Conference on Learning Representations*.

[98] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020a). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

[99] Lewis, P., Ott, M., Du, J., and Stoyanov, V. (2020b). Pretrained language models for biomedical and clinical tasks: Understanding and extending the state-of-the-art. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 146–157, Online. Association for Computational Linguistics.

[100] Li, B. and Gaussier, E. (2012). An information-based cross-language information retrieval model. In Baeza-Yates, R., de Vries, A. P., Zaragoza, H., Cambazoglu, B. B., Murdock, V., Lempel, R., and Silvestri, F., editors, *Advances in Information Retrieval*, pages 281–292, Berlin, Heidelberg. Springer Berlin Heidelberg.

[101] Li, Y., Wang, H., and Luo, Y. (2020). A Comparison of Pre-trained Vision-and-Language Models for Multimodal Representation Learning across Medical Images and Reports. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1999–2004, Seoul, Korea (South). IEEE.

[102] Lin, J. (2019). The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum*, 52(2):40–51.

[103] Lindberg, D. A. B., Humphreys, B. L., and McCray, A. T. (1993). The Unified Medical Language System. *Yearbook of medical informatics*, 1:41–51.

[104] Liu, D., Gong, Y., Fu, J., Yan, Y., Chen, J., Lv, J., Duan, N., and Zhou, M. (2020). Tell me how to ask again: Question data augmentation with controllable rewriting in continuous space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5798–5810, Online. Association for Computational Linguistics.

[105] Longpre, S., Lu, Y., Tu, Z., and DuBois, C. (2019). An exploration of data augmentation and sampling techniques for domain-agnostic question answering. *CoRR*, abs/1912.02145.

[106] Longpre, S., Wang, Y., and DuBois, C. (2020). How effective is task-agnostic data augmentation for pretrained transformers? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4401–4411, Online.

[107] Loomba, S., de Figueiredo, A., Piatek, S. J., de Graaf, K., and Larson, H. J. (2021). Measuring the impact of COVID-19 vaccine misinformation on vaccination intent in the UK and USA. *Nature Human Behaviour*, 5(3):337–348.

[108] Lopez, L. E., Cruz, D. K., Cruz, J. C. B., and Cheng, C. (2020). Transformer-based end-to-end question generation. *CoRR*, abs/2005.01107.

[109] Luo, M., Saxena, S., Mishra, S., Parmar, M., and Baral, C. (2022a). BioTABQA: Instruction Learning for Biomedical Table Question Answering. arXiv:2207.02419 [cs].

[110] Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., and Liu, T.-Y. (2022b). BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6).

[111] Luo, Y., Yang, B., Xu, D., and Tian, L. (2022c). A survey: Complex knowledge base question answering. In *2022 IEEE 2nd International Conference on Information Communication and Software Engineering (ICICSE)*, pages 46–52.

[112] MacAvaney, S., Yates, A., Cohan, A., and Goharian, N. (2019). CEDR: Contextualized Embeddings for Document Ranking. *CoRR*, abs/1904.07094.

[113] Malkov, Y. A. and Yashunin, D. A. (2016). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *Computing Research Repository (CoRR)*, abs/1603.09320.

[114] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[115] Margatina, K., Vernikos, G., Barrault, L., and Aletras, N. (2021). Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic.

[116] McDonald, R., Brokos, G., and Androutsopoulos, I. (2018). Deep Relevance Ranking Using Enhanced Document-Query Interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860, Brussels, Belgium.

[117] Miao, Z., Li, Y., Wang, X., and Tan, W.-C. (2020). Snippext: Semi-supervised opinion mining with augmented data. In *Proceedings of The Web Conference 2020*, page 617–628, New York, NY, USA. Association for Computing Machinery.

[118] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[119] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013b). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

[120] Mitra, B. and Craswell, N. (2018). An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126.

[121] Möller, T., Reina, A., Jayakumar, R., and Pietsch, M. (2020). COVID-QA: A question answering dataset for COVID-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online. Association for Computational Linguistics.

[122] Monajatipoor, M., Rouhsedaghat, M., Li, L. H., Chien, A., Kuo, C. C. J., Scalzo, F., and Chang, K.-W. (2021). BERTHop: An Effective Vision-and-Language Model for Chest X-ray Disease Diagnosis. *ArXiv*.

[123] Moon, J. H., Lee, H., Shin, W., Kim, Y.-H., and Choi, E. (2022). Multi-Modal Understanding and Generation for Medical Images and Text via Vision-Language Pre-Training. *IEEE Journal of Biomedical and Health Informatics*, 26(12):6070–6080.

[124] Moussallem, D., Arcan, M., Ngomo, A. N., and Buitelaar, P. (2019). Augmenting neural machine translation with knowledge graphs. *CoRR*, abs/1902.08816.

[125] Munkhdalai, T. and Yu, H. (2016). Reasoning with Memory Augmented Neural Networks for Language Comprehension. *CoRR*, abs/1610.06454.

[126] Mutabazi, E., Ni, J., Tang, G., and Cao, W. (2021). A review on medical textual question answering systems based on deep learning approaches. *Applied Sciences*, 11(12).

[127] Nentidis, A., Bougiatiotis, K., Krithara, A., and Paliouras, G. (2020). Results of the seventh edition of the bioasq challenge. *CoRR*, abs/2006.09174.

[128] Nentidis, A., Katsimpras, G., Vandorou, E., Krithara, A., Gascó, L., Krallinger, M., and Paliouras, G. (2021a). Overview of bioasq 2021: The ninth bioasq challenge on large-scale biomedical semantic indexing and question answering. *CoRR*, abs/2106.14885.

[129] Nentidis, A., Krithara, A., Bougiatiotis, K., Krallinger, M., Penagos, C. R., Villegas, M., and Paliouras, G. (2021b). Overview of bioasq 2020: The eighth bioasq challenge on large-scale biomedical semantic indexing and question answering. *CoRR*, abs/2106.14618.

[130] Neves, M. (2014). An analysis on the entity annotations in biological corpora. *F1000Research*, 3:96.

[131] Neves, M. and Leser, U. (2014). A survey on annotation tools for the biomedical literature. *Briefings in Bioinformatics*, 15(2):327–340.

[132] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016.*

[133] Nogueira, R. and Cho, K. (2019). Passage Re-ranking with BERT. *CoRR*, abs/1901.04085.

[134] Nugraha, H. S. and Suyanto, S. (2019). Typographic-based data augmentation to improve a question retrieval in short dialogue system. In *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 44–49.

[135] Onishi, T., Wang, H., Bansal, M., Gimpel, K., and McAllester, D. (2016). Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235. Association for Computational Linguistics.

[136] Pagliardini, M., Gupta, P., and Jaggi, M. (2018). Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 528–540.

[137] Pal, A., Umapathi, L. K., and Sankarasubbu, M. (2022). Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Flores, G., Chen, G. H., Pollard, T., Ho, J. C., and Naumann, T., editors, *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

[138] Pampari, A., Raghavan, P., Liang, J., and Peng, J. (2018). emrQA: A Large Corpus for Question Answering on Electronic Medical Records. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2357–2368, Brussels, Belgium.

[139] Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., and Cheng, X. (2017). DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*.

[140] Pappas, D. and Androutsopoulos, I. (2021). A neural model for joint document and snippet ranking in question answering for large document collections. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3896–3907, Online. Association for Computational Linguistics.

[141] Pappas, D., Androutsopoulos, I., and Papageorgiou, H. (2018). BioRead: A new dataset for biomedical reading comprehension. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

[142] Pappas, D., Malakasiotis, P., and Androutsopoulos, I. (2022). Data augmentation for biomedical factoid question answering. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 63–81, Dublin, Ireland. Association for Computational Linguistics.

[143] Pappas, D., McDonald, R., Brokos, G.-I., and Androutsopoulos, I. (2019). AUEB at BioASQ 7: document and snippet retrieval. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 607–623, Wurzburg, Germany.

[144] Pappas, D., Stavropoulos, P., and Androutsopoulos, I. (2020a). AUEB-NLP at BioASQ 8: Biomedical Document and Snippet Retrieval. In *Proceedings of the 8th BioASQ workshop at the Conference and Labs of the Evaluation Forum (CLEF 2020)*, pages 41–54, Thessaloniki, Greece.

[145] Pappas, D., Stavropoulos, P., Androutsopoulos, I., and McDonald, R. (2020b). BioMRC: A dataset for biomedical machine reading comprehension. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 140–149, Online. Association for Computational Linguistics.

[146] Parmar, P., Ryu, J., Pandya, S., Sedoc, J., and Agarwal, S. (2022). Health-focused conversational agents in person-centered care: a review of apps. *npj Digital Medicine*, 5(1):1–9.

[147] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[148] Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[149] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, Louisiana.

[150] Qin, T., Liu, T.-Y., Xu, J., and Li, H. (2010). Letor: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retrieval*.

[151] Qiu, J. and Xiong, D. (2019). Generating highly relevant questions. *CoRR*, abs/1910.03401.

[152] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

[153] Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

[154] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

[155] Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

[156] Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

[157] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. (1994). Okapi at TREC-3. In Harman, D. K., editor, *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST).

[158] Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *ArXiv*, abs/2002.12327.

[159] Roozenbeek, J., Schneider, C. R., Dryhurst, S., Kerr, J., Freeman, A. L. J., Recchia, G., van der Bles, A. M., and van der Linden, S. (2020). Susceptibility to misinformation about COVID-19 around the world. *Royal Society Open Science*, 7(10):201199.

[160] Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

[161] Safran, M., Rosen, N., Twik, M., BarShir, R., Stein, T. I., Dahary, D., Fishilevich, S., and Lancet, D. (2021). *The GeneCards Suite*, pages 27–56. Springer Nature Singapore, Singapore.

[162] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

[163] Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[164] Sennrich, R., Haddow, B., and Birch, A. (2016a). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany.

[165] Sennrich, R., Haddow, B., and Birch, A. (2016b). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

[166] Seo, M. J., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2016). Bidirectional Attention Flow for Machine Comprehension. *CoRR*, abs/1611.01603.

[167] Severyn, A. and Moschitti, A. (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*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

[168] Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

[169] Shen, D., Zheng, M., Shen, Y., Qu, Y., and Chen, W. (2020). A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *ArXiv*, abs/2009.13818.

[170] Shin, H.-C., Zhang, Y., Bakhturina, E., Puri, R., Patwary, M., Shoeybi, M., and Mani, R. (2020). BioMegatron: Larger biomedical domain language model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4700–4706, Online. Association for Computational Linguistics.

[171] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(60).

[172] Shorten, C., Khoshgoftaar, T. M., and Furht, B. (2021). Text data augmentation for deep learning. *Journal of Big Data*, 8(101).

[173] Simmons, R. F. (1965). Answering english questions by computer: A survey. *Commun. ACM*, 8(1):53–70.

[174] Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., Payne, P., Seneviratne, M., Gamble, P., Kelly, C., Scharli, N., Chowdhery, A., Mansfield, P., Arcas, B. A. y., Webster, D., Corrado, G. S., Matias, Y., Chou, K., Gottweis, J., Tomasev, N., Liu, Y., Rajkomar, A., Barral, J., Semturs, C., Karthikesalingam, A., and Natarajan, V. (2022). Large language models encode clinical knowledge.

[175] Song, B., Li, F., Liu, Y., and Zeng, X. (2021). Deep learning methods for biomedical named entity recognition: a survey and qualitative comparison. *Briefings in Bioinformatics*, 22(6):bbab282.

[176] Soysal, E., Wang, J., Jiang, M., Wu, Y., Pakhomov, S., Liu, H., and Xu, H. (2017). CLAMP – a toolkit for efficiently building customized clinical natural language processing pipelines. *Journal of the American Medical Informatics Association*, 25(3):331–336.

[177] Spitkovsky, V. I. and Chang, A. X. (2012). A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

[178] Suggu, S. P., Goutham, K. N., Chinnakotla, M. K., and Shrivastava, M. (2016). Deep feature fusion network for answer quality prediction in community question answering. *CoRR*, abs/1606.07103.

[179] Sultan, M. A., Chandel, S., Fernandez Astudillo, R., and Castelli, V. (2020). On the importance of diversity in question generation for QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656, Online. Association for Computational Linguistics.

[180] Šuster, S. and Daelemans, W. (2018). CliCR: a Dataset of Clinical Case Reports for Machine Reading Comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1551–1563, New Orleans, Louisiana.

[181] Tinn, R., Cheng, H., Gu, Y., Usuyama, N., Liu, X., Naumann, T., Gao, J., and Poon, H. (2021). Fine-tuning large neural language models for biomedical natural language processing. *CoRR*, abs/2112.07869.

[182] Tohti, T., Abdurxit, M., and Hamdulla, A. (2022). Medical QA Oriented Multi-Task Learning Model for Question Intent Classification and Named Entity Recognition. *Information*, 13(12):581.

[183] Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. (2017). NewsQA: A Machine Comprehension Dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada.

[184] Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M., Weissenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D., Almirantis, Y., Pavlopoulos, J., Baskiotis, N., Gallinari, P., Artieres, T., Ngonga, A., Heino, N., Gaussier, E., Barrio-Alvers, L., Schroeder, M., Androutsopoulos, I., and Paliouras, G. (2015). An overview of the BioASQ Large-Scale Biomedical Semantic Indexing and Question Answering Competition. *BMC Bioinformatics*, 16(138).

[185] Vachev, K., Hardalov, M., Karadzhov, G., Georgiev, G., Koychev, I., and Nakov, P. (2021). Generating answer candidates for quizzes and answer-aware question generators. In *Proceedings of the Student Research Workshop Associated with RANLP 2021*, pages 203–209, Online. INCOMA Ltd.

[186] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

[187] Voorhees, E. M. (2001). The TREC question answering track. *Natural Language Engineering*, 7(4):361–378.

[188] Voorhees, E. M. and Harman, D. K., editors (1999). *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*, volume Special Publication 500-246. National Institute of Standards and Technology (NIST).

[189] Wallace, B. C., Small, K., Brodley, C. E., Lau, J., and Trikalinos, T. A. (2012). Deploying an interactive machine learning system in an evidence-based practice center: Abstrackr. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, IHI '12, page 819–824, New York, NY, USA. Association for Computing Machinery.

[190] Wang, D. and Nyberg, E. (2015). A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 707–712. Association for Computational Linguistics.

[191] Wang, L. L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R., Liu, Z., Merrill, W., Mooney, P., Murdick, D., Rishi, D., Sheehan, J., Shen, Z., Stilson, B., Wade, A. D., Wang, K., Wilhelm, C., Xie, B., Raymond, D., Weld, D. S., Etzioni, O., and Kohlmeier, S. (2020). Cord-19: The covid-19 open research dataset.

[192] Wang, W. Y. and Yang, D. (2015). That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

[193] Wei, C.-H., Harris, B. R., Li, D., Berardini, T. Z., Huala, E., Kao, H.-Y., and Lu, Z. (2012). Accelerating literature curation with text-mining tools: a case study of using PubTator to curate genes in PubMed abstracts. *Database*, 2012.

[194] Wei, J. and Zou, K. (2019). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

[195] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G. S., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

[196] Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2020). Unsupervised data augmentation for consistency training. *arXiv*.

[197] Xiong, C., Dai, Z., Callan, J., Liu, Z., and Power, R. (2017). End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64, Shinjuku, Tokyo, Japan.

[198] Xu, G., Rong, W., Wang, Y., Ouyang, Y., and Xiong, Z. (2021a). External features enriched model for biomedical question answering. *BMC Bioinformatics*, 22(1):272.

[199] Xu, L., Sanders, L., Li, K., and Chow, J. C. L. (2021b). Chatbot for Health Care and Oncology Applications Using Artificial Intelligence and Machine Learning: Systematic Review. *JMIR Cancer*, 7(4):e27850.

[200] Yan, Y., Yin, X.-C., Zhang, B.-W., Yang, C., and wei Hao, H. (2016). Semantic indexing with deep learning: a case study. *Big Data Analytics*, 1:1–13.

[201] Yang, C. C. (2010). Search engines information retrieval in practice. *Journal of the American Society for Information Science and Technology*, 61(2):430–430.

[202] Yang, L., Ai, Q., Guo, J., and Croft, W. B. (2016). anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 287–296.

[203] Yang, L., Zhang, M., Li, C., Bendersky, M., and Najork, M. (2020a). Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching. In *CIKM*.

[204] Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., and Lin, J. (2019a). End-to-End open-Domain Question Answering with BERTserini. *CoRR*, abs/1902.01718.

[205] Yang, W., Xie, Y., Tan, L., Xiong, K., Li, M., and Lin, J. (2019b). Data augmentation for bert fine-tuning in open-domain question answering. *ArXiv*, abs/1904.06652.

[206] Yang, W., Zhang, H., and Lin, J. (2019c). Simple Applications of BERT for Ad Hoc Document Retrieval. *CoRR*, abs/1903.10972.

[207] Yang, X., PourNejatian, N., Shin, H. C., Smith, K. E., Parisien, C., Compas, C., Martin, C., Flores, M. G., Zhang, Y., Magoc, T., Harle, C. A., Lipori, G., Mitchell, D. A., Hogan, W. R., Shenkman, E. A., Bian, J., and Wu, Y. (2022). Gatortron: A large language model for clinical natural language processing. *medRxiv*.

[208] Yang, Y., Malaviya, C., Fernandez, J., Swayamdipta, S., Le Bras, R., Wang, J.-P., Bhagavatula, C., Choi, Y., and Downey, D. (2020b). Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics.

[209] Yang, Y., Yih, W.-t., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018. Association for Computational Linguistics.

[210] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. (2018). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380. Association for Computational Linguistics.

[211] Yasunaga, M., Bosselut, A., Ren, H., Zhang, X., Manning, C. D., Liang, P., and Leskovec, J. (2022). Deep bidirectional language-knowledge graph pretraining.

[212] Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016). ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4.

[213] Yoon, W., Lee, J., Kim, D., Jeong, M., and Kang, J. (2020). Pre-trained language model for biomedical question answering. In Cellier, P. and Driessens, K., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 727–740, Cham. Springer International Publishing.

[214] Yuan, H., Yuan, Z., Gan, R., Zhang, J., Xie, Y., and Yu, S. (2022). BioBART: Pretraining and evaluation of a biomedical generative language model. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 97–109, Dublin, Ireland. Association for Computational Linguistics.

[215] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33.

[216] Zaib, M., Zhang, W. E., Sheng, Q. Z., Mahmood, A., and Zhang, Y. (2022). Conversational question answering: a survey. *Knowledge and Information Systems*, 64(12):3151–3195.

[217] Zamani, H., Dehghani, M., Croft, W. B., Learned-Miller, E., and Kamps, J. (2018). From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 497–506, New York, NY, USA. Association for Computing Machinery.

[218] Zhai, C. and Lafferty, J. (2001a). Language model based on the jelinek-mercer smoothing method. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '01)*, pages 334–342, New York, NY, USA. Association for Computing Machinery.

[219] Zhai, C. and Lafferty, J. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 334–342, New York, NY, USA. Association for Computing Machinery.

[220] Zhang, C., Lai, Y., Feng, Y., and Zhao, D. (2021). A review of deep learning in question answering over knowledge bases. *AI Open*, 2:205–215.

[221] Zhang, S. and Bansal, M. (2019). Addressing semantic drift in question generation for semi-supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2495–2509, Hong Kong, China. Association for Computational Linguistics.

[222] Zhang, S., Zhang, X., Wang, H., Cheng, J., Li, P., and Ding, Z. (2017). Chinese medical question answer matching using end-to-end character-level multi-scale cnns. *Applied Sciences*, 7(8).

[223] Zhang, X., Wu, J., He, Z., Liu, X., and Su, Y. (2018). Medical Exam Question Answering with Large-scale Reading Comprehension. *ArXiv*.

[224] Zhang, Z., jie Yang, J., and Zhao, H. (2020). Retrospective Reader for Machine Reading Comprehension. *ArXiv*.

[225] Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., and Chua, T. (2021). Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774.

[226] Zhu, H., Paschalidis, I. C., and Tahmasebi, A. (2018a). Clinical concept extraction with contextual word embedding. *arXiv*.

[227] Zhu, M., Ahuja, A., Juan, D.-C., Wei, W., and Reddy, C. K. (2020). Question answering with long multiple-span answers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3840–3849, Online. Association for Computational Linguistics.

[228] Zhu, P., Zhang, Z., Li, J., Huang, Y., and Zhao, H. (2018b). Lingke: a fine-grained multi-turn chatbot for customer service. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 108–112, Santa Fe, New Mexico. Association for Computational Linguistics.

[229] Zou, J., Thoma, G., and Antani, S. (2020). Unified deep neural network for segmentation and labeling of multipanel biomedical figures. *Journal of the Association for Information Science and Technology*, 71(11):1327–1340.

# Appendix A

# Detailed experiments

## A.1 Details on Data Augmentation experiments

In this section, we provide more details on the data augmentation experiments that we conducted as part of our research. We present the results of these experiments and discuss the implications of our findings for the use of data augmentation in biomedical factoid question answering.

### A.1.1 BIOASQ additional experiments

#### Learning Curves

In Tables A.1, A.2, A.3 we report the performance of the factoid QA model described in Section 6.3.1 when a limited number of additional data can be created. In these experiments, we use the same data split that we used in Table 6.2 and apply data augmentation using BIOMRC, WORD2VEC, and BIOLM. We use four different sample sizes (10K, 30K, 50K, and 100K additional training instances) and train our model several times gradually increasing the training size.

When using BIOMRC for data augmentation, it is preferable to use 100K training examples to pre-train the model and then further finetune it with the original BIOASQ data.

The same can be observed when using WORD2VEC for data augmentation where the best DEV PRAUC is achieved (95.91). On the other hand, even when 10K augmented training examples are used the model achieves almost identical results (95.19 DEV PRAUC). On unseen data however we observe that as we add more augmented data, the performance of the models increases (from 83.65 to 89.93 DEV PRAUC). It is probable that using more data allows generalization of the model hence the better performance on unseen data.

| Method | Setting | + train ex. | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|---|---|
| No AUG | N/A | 2848 | 0.8957 | 0.5924 | 0.7678 | 0.6000 |
| BIOMRC | PRETRAIN | 10000 | 0.9120 | 0.7758 | 0.7928 | 0.6191 |
| BIOMRC | FINETUNE | 12848 | 0.9235 | 0.7550 | 0.8146 | 0.6776 |
| BIOMRC | COMB | 12848 | 0.9315 | 0.7951 | 0.8204 | 0.6748 |
| BIOMRC | PRETRAIN | 30000 | 0.9057 | 0.7541 | 0.8012 | 0.6462 |
| BIOMRC | FINETUNE | 32848 | 0.9232 | 0.7590 | 0.8154 | 0.6994 |
| BIOMRC | COMB | 32848 | 0.9219 | 0.7491 | 0.8205 | 0.6462 |
| BIOMRC | PRETRAIN | 50000 | 0.9119 | 0.7634 | 0.8223 | 0.6591 |
| BIOMRC | FINETUNE | 52848 | 0.9183 | 0.7648 | 0.8023 | 0.6896 |
| BIOMRC | COMB | 52848 | 0.9151 | 0.7808 | **0.8383** | 0.6749 |
| BIOMRC | PRETRAIN | 100000 | 0.9093 | 0.7844 | 0.8149 | 0.6246 |
| BIOMRC | FINETUNE | 102848 | **0.9321** | 0.7860 | 0.8266 | 0.6987 |
| BIOMRC | COMB | 102848 | 0.9239 | 0.8010 | 0.8124 | 0.6308 |

Table A.1 results for BIOASQ-8 learning curve using BIOMRC

| Method | Setting | + train ex. | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|---|---|
| No AUG | N/A | 2848 | 0.8957 | 0.5924 | 0.7678 | 0.6000 |
| WORD2VEC | PRETRAIN | 10000 | 0.9559 | 0.7448 | 0.8359 | 0.6821 |
| WORD2VEC | FINETUNE | 12848 | 0.9519 | 0.7941 | 0.8365 | 0.6809 |
| WORD2VEC | COMB | 12848 | 0.9560 | 0.7654 | 0.8499 | 0.6860 |
| WORD2VEC | PRETRAIN | 30000 | 0.9528 | 0.7425 | 0.8310 | 0.6834 |
| WORD2VEC | FINETUNE | 32848 | 0.9418 | 0.8165 | 0.8501 | 0.7109 |
| WORD2VEC | COMB | 32848 | 0.9520 | 0.7678 | 0.8408 | 0.6924 |
| WORD2VEC | PRETRAIN | 50000 | 0.9516 | 0.7540 | 0.8301 | 0.6796 |
| WORD2VEC | FINETUNE | 52848 | 0.9489 | 0.8200 | 0.8496 | 0.7276 |
| WORD2VEC | COMB | 52848 | 0.9513 | 0.7540 | 0.8386 | 0.6770 |
| WORD2VEC | PRETRAIN | 100000 | 0.9536 | 0.7540 | 0.8117 | 0.6788 |
| WORD2VEC | FINETUNE | 102848 | **0.9591** | 0.8027 | **0.8993** | 0.7804 |
| WORD2VEC | COMB | 102848 | 0.9522 | 0.7678 | 0.8333 | 0.6745 |

Table A.2 results for BIOASQ-8 learning curve using WORD2VEC

The use of BIOLM for data augmentation yielded comparable results to the use of WORD2VEC for data augmentation in terms of performance on the development set. No significant differences in scores were observed across different settings within the development set. The best DEV PRAUC was achieved when augmented and original data were shuffled and then used to train the model (94.45 DEV PRAUC). Contrary to using WORD2VEC, using more augmented data produced by BIOLM reduces the TEST PRAUC.

In further experiments, we simulated the scenario where fewer original training examples are available, by using only a portion of the data of the original BIOASQ dataset. Each time we sample a subset of the dataset (20%, 40%, 60%, 80%, and 100%) and use this portion of the dataset to train, apply data augmentation and fine-tune the model. In Table A.4 we can see the results for data augmentation using BIOLM. To save time, we used only one data augmentation method in this experiment, and we selected BIOLM as it performed best for the

| Method | Setting | + train ex. | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|---|---|
| No AUG | N/A | 2848 | 0.8957 | 0.5924 | 0.7678 | 0.6000 |
| BIOLM | PRETRAIN | 10000 | 0.9406 | 0.8056 | **0.8417** | 0.6751 |
| BIOLM | FINETUNE | 12848 | 0.9406 | 0.8021 | 0.8346 | 0.6783 |
| BIOLM | COMB | 12848 | 0.9406 | 0.8177 | 0.8235 | 0.6751 |
| BIOLM | PRETRAIN | 30000 | 0.9363 | 0.7337 | 0.8374 | 0.6859 |
| BIOLM | FINETUNE | 32848 | 0.9369 | 0.8211 | 0.8363 | 0.6507 |
| BIOLM | COMB | 32848 | 0.9375 | 0.7763 | 0.8323 | 0.6731 |
| BIOLM | PRETRAIN | 50000 | 0.9394 | 0.7993 | 0.8229 | 0.6636 |
| BIOLM | FINETUNE | 52848 | 0.9366 | 0.8200 | 0.8247 | 0.6636 |
| BIOLM | COMB | 52848 | **0.9445** | 0.8039 | 0.8276 | 0.6584 |
| BIOLM | PRETRAIN | 100000 | 0.9379 | 0.8154 | 0.8223 | 0.6674 |
| BIOLM | FINETUNE | 102848 | 0.9381 | 0.8090 | 0.8272 | 0.6769 |
| BIOLM | COMB | 102848 | 0.9384 | 0.7740 | 0.8262 | 0.6526 |

Table A.3 results for BIOASQ-8 learning curve using BIOLM

COVIDQA dataset and achieved the second-best performance for the BIOASQ dataset. Overall data augmentation always improves PRAUC performance even if we only use 20% of the original BIOASQ dataset. The best performance is achieved when the entire original dataset is used to create augmented data. In this case, even using just the augmented data to pre-train the model achieves the best DEV and TEST PRAUC scores (91.18 and 91.16 respectively).

### 10-fold cross-validation

10-fold cross-validation is a common method used in natural language processing (NLP) and other fields to evaluate the performance of a machine learning model. It involves dividing the dataset into 10 equal-sized folds or partitions. The process is repeated 10 times, with each fold serving as the test set once and the training set of each repetition being the other 9 folds. The performance of the model is then averaged across all 10 iterations to give a final evaluation score. Cross-validation helps to assess the model's generalization ability, or how well it performs on unseen data, and reduces the variance in the evaluation score by training and evaluating the model on different subsets of the data.

In experiments conducted on BIOASQ 8 data we observe that all data augmentation techniques improve the results of the model (Table A.5). In 10-fold cross-validation, the best DEV PRAUC was achieved by increasing the context of the processed snippet (79.27 PRAUC). For unseen data, however, the best score was once more achieved using WORD2VEC for data augmentation (82.17). Further improvement could be achieved by combining the two best data augmentation techniques, i.e. first increase the context with surrounding sentences and then replace tokens using WORD2VEC on the augmented text.

| Method | Setting | data percentage | + train ex. | best ep. | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|---|---|---|---|
| No AUG | N/A | 20% (572) | 0 | 46 | 84.93 | 39.28 | 82.22 | 35.37 |
| No AUG | N/A | 40% (1,145) | 0 | 48 | 88.60 | 52.91 | 83.41 | 49.24 |
| No AUG | N/A | 60% (1,717) | 0 | 42 | 89.62 | 53.18 | 83.34 | 49.63 |
| No AUG | N/A | 80% (2,290) | 0 | 26 | 89.42 | 52.22 | 83.13 | 49.01 |
| No AUG | N/A | 100% (2,863) | 0 | 19 | 89.49 | 48.60 | 83.49 | 45.37 |
| BIOLM | PRETRAIN | 0 | 2,999 | 11 | 88.47 | 75.23 | 86.24 | 80.17 |
| BIOLM | PRETRAIN | 0 | 6,052 | 13 | 89.36 | 79.64 | 88.53 | 81.94 |
| BIOLM | PRETRAIN | 0 | 9,106 | 11 | 89.66 | 81.61 | 88.45 | 79.81 |
| BIOLM | PRETRAIN | 0 | 11,736 | 10 | 90.58 | 81.56 | 88.67 | 75.82 |
| BIOLM | PRETRAIN | 0 | 14,417 | 43 | **91.18** | 82.97 | **91.16** | 79.97 |
| BIOLM | FINETUNE | 20% (572) | 2,999 | 8 | 87.75 | 79.96 | 87.27 | 81.08 |
| BIOLM | FINETUNE | 40% (1,145) | 6,052 | 7 | 88.61 | 81.00 | 88.55 | **82.68** |
| BIOLM | FINETUNE | 60% (1,717) | 9,106 | 7 | 89.64 | 81.54 | 89.34 | 81.57 |
| BIOLM | FINETUNE | 80% (2,290) | 11,736 | 15 | 90.56 | 82.76 | 87.41 | 78.01 |
| BIOLM | FINETUNE | 100% (2,863) | 14,417 | 1 | 90.92 | 81.11 | 90.00 | 79.48 |
| BIOLM | COMB | 20% (572) | 2,999 | 10 | 88.62 | 77.04 | 86.24 | 80.17 |
| BIOLM | COMB | 40% (1,145) | 6,052 | 12 | 89.35 | 79.30 | 88.54 | 80.93 |
| BIOLM | COMB | 60% (1,717) | 9,106 | 23 | 90.24 | **83.02** | 89.77 | 81.78 |
| BIOLM | COMB | 80% (2,290) | 11,736 | 9 | 90.59 | 81.34 | 88.67 | 75.82 |
| BIOLM | COMB | 100% (2,863) | 14,417 | 7 | 90.38 | 78.18 | 87.02 | 78.13 |

Table A.4 results for BIOASQ-8 learning curve 2

### A.1.2 COVIDQA additional experiments

We also applied 10-fold cross-validation to COVIDQA data and report the results in Table A.6. Once again the replacement of tokens using a form of language model (in our experiments we used BIOLM) seems to be the best method for data augmentation. Overall the DEV PRAUC score is improved by 8 points (from 58.47 to 66.79). However, similarly to BIOASQ the best method for unseen data seems to be the WORD2VEC technique, which achieves 61.74 TEST PRAUC.

Further improvement can be observed when using the best performing model for the BIOASQ dataset as a starting point and further fine-tuning the model using original and augmented data from the COVIDQA dataset. Just using the best performing BIOASQ model (Table A.7, 2nd row) achieves a DEV PRAUC score 67.97 and achieves the best overall TEST PRAUC (45.60). Further fine-tuning of the model using augmented data from the COVIDQA dataset improves performance on the development set, but the PRAUC score on unseen data does not improve. Overall using the best BIOASQ model always improves the results on the COVIDQA dataset. It might be the case that BIOASQ data are more diverse and allow more generalization. Fine-tuning the BIOASQ model on COVIDQA data leads to indirect over-fitting (through tuning hyper-parameters) of the model on development data.

| Method | Setting | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|---|
| No AUG | N/A | 0.7462 | 0.3482 | 0.7719 | 0.3508 |
| CONTEXT | PRETRAIN | **0.7927** | 0.6085 | 0.8117 | 0.6386 |
| CONTEXT | FINETUNE | 0.7874 | **0.6634** | 0.8173 | 0.6775 |
| CONTEXT | COMB | 0.7893 | 0.5440 | 0.8109 | 0.6367 |
| BIOMRC | PRETRAIN | 0.7471 | 0.4432 | 0.7355 | 0.4387 |
| BIOMRC | FINETUNE | 0.7697 | 0.5524 | 0.7985 | 0.5833 |
| BIOMRC | COMB | 0.7724 | 0.5266 | 0.7913 | 0.5880 |
| BTR | PRETRAIN | 0.7769 | 0.6234 | 0.7999 | 0.6692 |
| BTR | FINETUNE | 0.7757 | 0.6429 | 0.8078 | 0.6796 |
| BTR | COMB | 0.7751 | 0.6135 | 0.8083 | 0.6655 |
| WORD2VEC | PRETRAIN | 0.7902 | 0.6006 | 0.8002 | 0.6175 |
| WORD2VEC | FINETUNE | 0.7833 | 0.6530 | **0.8217** | 0.6681 |
| WORD2VEC | COMB | 0.7903 | 0.6068 | 0.8041 | 0.6394 |
| BIOLM | PRETRAIN | 0.7871 | 0.6282 | 0.8052 | 0.6800 |
| BIOLM | FINETUNE | 0.7793 | 0.6449 | 0.8209 | **0.6912** |
| BIOLM | COMB | 0.7861 | 0.6213 | 0.8075 | 0.6769 |
| T5 | PRETRAIN | 0.7339 | 0.4442 | 0.7602 | 0.4503 |
| T5 | FINETUNE | 0.7650 | 0.5660 | 0.7927 | 0.5982 |
| T5 | COMB | 0.7562 | 0.5248 | 0.7731 | 0.5254 |

Table A.5 10-fold cross-validation results for BIOASQ-8 (2021).

| Method | Setting | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|---|
| No AUG | N/A | 0.5847 | 0.0647 | 0.5511 | 0.0682 |
| CONTEXT | PRETRAIN | 0.6312 | 0.0869 | 0.5563 | 0.0862 |
| CONTEXT | FINETUNE | 0.647 | 0.2797 | 0.6078 | 0.2715 |
| CONTEXT | COMB | 0.6421 | 0.1324 | 0.562 | 0.0914 |
| BIOMRC | PRETRAIN | 0.6161 | 0.1799 | 0.5526 | 0.1592 |
| BIOMRC | FINETUNE | 0.6149 | 0.2169 | 0.5992 | 0.2215 |
| BIOMRC | COMB | 0.6169 | 0.1714 | 0.5552 | 0.1538 |
| WORD2VEC | PRETRAIN | 0.64 | 0.1915 | 0.5952 | 0.1667 |
| WORD2VEC | FINETUNE | 0.651 | 0.2885 | 0.6101 | 0.2713 |
| WORD2VEC | COMB | 0.6517 | 0.2229 | **0.6174** | 0.2056 |
| BIOLM | PRETRAIN | 0.6644 | 0.3246 | 0.5976 | 0.2575 |
| BIOLM | FINETUNE | 0.6509 | **0.3634** | 0.6085 | **0.2947** |
| BIOLM | COMB | **0.6679** | 0.3064 | 0.5965 | 0.2535 |

Table A.6 10-fold cross-validation results for COVIDQA.

| Method | Setting | PRAUC (dev) | F1 0.5 (dev) | PRAUC (test) | F1 0.5 (test) |
|---|---|---|---|---|---|
| No AUG | N/A | 0.5847 | 0.0647 | 0.5511 | 0.0682 |
| BEST-BIOASQ | N/A | 0.6797 | **0.4614** | **0.6757** | **0.4560** |
| BEST-BIOASQ + WORD2VEC | PRETRAIN | 0.6273 | 0.1975 | 0.6170 | 0.1961 |
| BEST-BIOASQ + WORD2VEC | FINETUNE | 0.6627 | 0.3855 | 0.6197 | 0.3538 |
| BEST-BIOASQ + WORD2VEC | COMB | 0.6474 | 0.2669 | 0.6422 | 0.2526 |
| BEST-BIOASQ + BIOLM | PRETRAIN | 0.6775 | 0.3986 | 0.6604 | 0.3754 |
| BEST-BIOASQ + BIOLM | FINETUNE | 0.6570 | 0.3662 | 0.6581 | 0.3577 |
| BEST-BIOASQ + BIOLM | COMB | **0.6838** | 0.4001 | 0.6567 | 0.3787 |

Table A.7 10-fold cross-validation results for COVIDQA.

# Appendix B

# Dense Retrieval

The present appendix represents an extension of the work described in Section 4.6.1. Our aim is to extend the study by directly indexing sentence embeddings in order to retrieve sentences, rather than first retrieving documents and then retrieving sentences within the documents. This line of work is an extension of the SEMISER study (Section 4.6.1), however, further exploration was not pursued as it resulted in poor results in the BIOASQ 7.

## B.1 Introduction

Sentence Retrieval on huge corpora is a difficult task. The most common way to detect relevant sentences inside a huge set of documents consists of mainly three steps:

1. retrieve and rank relevant documents,
2. retrieve and rank relevant paragraphs,
3. retrieve and rank relevant sentences.

These steps suffer from the cumulative error which is added through steps 1 to 3. Even when the output of step 2 is a set of truly relevant paragraphs, only a small portion of the contained sentences is usually relevant.

Dense Retrieval involves using neural networks to retrieve relevant documents given a query. The approach seeks to embed both the query and the document into a dense vector representation, which is then compared to determine the relevance of the document. The importance of Dense Retrieval using neural networks in the NLP field lies in its ability to handle large-scale datasets, as well as its ability to capture and exploit more nuanced relationships between queries and documents. Unlike traditional retrieval engines such as BM25, which rely on simple heuristics to rank documents, Dense Retrieval using neural

networks is able to learn more sophisticated ranking functions from the data. In addition, Dense Retrieval can be fine-tuned to different domains and datasets, making it highly flexible and adaptable to a wide range of use cases.

In this chapter, we propose new methods for dense retrieval using sentence and query embeddings. Our deep learning models are trained so that the embedding of a question and the embeddings of sentences that answer the question are similar. We focus on creating a neural network-based index of sentence embeddings where millions of pre-computed sentence embeddings are stored, therefore cross-attention mechanisms between a sentence and a question cannot be applied. When a query is submitted our model extracts the embedding of the question and detects the most similar embeddings that are pre-calculated and indexed in our database.

The models' inputs are word vector representations that can also be pre-trained on larger corpora (as when using GloVe [148] or Word2Vec [118]). We create sentence embeddings in a way that sentences containing the same n-grams should have similar vectors in the embedding space. Finally, we extract the data to train our model in an unsupervised way so that we could create an unlimited amount of training instances given some texts. The extracted sentence embeddings can be indexed in a database so that a simple similarity function (e.g. cosine similarity) applied to pairs of a vector representation of a sentence and a vector representation of a query could return relevant sentences. The retrieval can be fast and the required memory can be adjusted by adjusting the sentence embeddings' size. Our training instances do not depend on sequences of sentences and our sentence embeddings do not depend on neighboring sentences, making it easy to create embeddings for sentences without a context, which allows our approach to be used on shorter text spans.

We create our sentence embeddings for retrieval trying to solve an auxiliary task that is trivial for humans and can be solved by simple computations, i.e. given a sentence embedding and an n-gram embedding the model must decide whether the n-gram can be found in the sentence or not. Even though this task is trivial for humans and not very challenging for deep learning methods, it has the potential to create really useful sentence embeddings as a byproduct given that the representations of sentences with similar meaning should have representations with high cosine similarity. For example given the two sentences: *'a major breakthrough in AI .'* and *'an innovation in artificial intelligence .'* we would expect the embeddings of these two sentences to be close in the semantic space. Furthermore, we would like a mechanism that given the n-gram *'artificial intelligence'* as a query would return both sentences in the results even if the n-gram cannot be found in the first sentence.

To understand the intuition behind our models we present the following sentences as an example:

- S1: Machine learning is generating new opportunities for innovative research in energy economics and finance.
- S2: Machine learning
- S3: Deep learning is generating new opportunities for innovative research in energy economics and finance.
- S4: Therefore, machine learning is a powerful tool for economists.
- S5: Machine learning is generating new opportunities for innovative research in energy economics and finance, therefore, machine learning is a powerful tool for economists.

We can develop a model that creates vector representations for each sentence. We can also assume that the model only assigns zeros and ones to the vector representation like the multi-hot vector representations of sentences presented in Figure B.1. Similarly, a traditional database creates an inverted index with vectors as big as the vocabulary. Given a text, an analyzer creates a vector representation as big as the length of the vocabulary. Then it creates the text's multi-hot vector representation.

Given a vocabulary set $V = w_1, w_2, ..., w_n$, where $n$ is the size of the vocabulary, and a document $d = w_{d_1}, w_{d_2}, ..., w_{d_m}$, where $m$ is the number of words in the document, the multi-hot vector representation of $d$ is a binary vector $x$ of length $n$, where each element $x_i$ represents the presence or absence of the word $w_i$ in the document $d$.

Mathematically, the multi-hot vector representation of $d$ can be defined as:

$$x_i = \begin{cases} 1 & \text{if } w_i \in d \\ \\ 0 & \text{otherwise} \end{cases} \tag{B.1}$$

Traditional databases however cannot automatically detect similar entities in the sentences (like 'machine learning' and 'deep learning' in the examples). Ideally, we could train a deep learning model that would detect similar entities (or concepts) in sentences and assign the value 1 to the multi-hot vector if an entity (or concept) is present in the sentence. These models would create vectors that comply with the following rules:

1. $\|Vec(S2)\| < \|Vec(S1)\|$, if $S2$ is a part of $S1$.
2. $\|Vec(S1) - Vec(S3)\| \simeq 0$,
   since the concept of 'deep learning' is close to the concept of 'machine learning' and the rest of the sentence remains the same.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Vocabulary →<br>Sentences ↓** | **machine** | **learning** | **is** | **generating** | **new** | **opportunities** | **for** | **innovative** | **research** | **in** | **energy** | **economics** | **and** | **finance** | **deep** | **therefore** | **a** | **powerful** | **tool** | **economists** |
| **S1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **S2** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **S3** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **S4** | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| **S5** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

Fig. B.1 Example of multi-hot embedding of sentences.

3. $\|Vec(S2) - Vec(S1)\| < \|Vec(S1) - Vec(S_{rand})\|$,
   if $S2$ is a part of $S1$ and $S_{rand}$ is a random sentence.

4. $\|Vec(S2) - Vec(S1)\| < \|Vec(S2) - Vec(S_{rand})\|$,
   if $S2$ is a part of $S1$ and $S_{rand}$ is a random sentence.

5. $\|Vec(S5)\| \simeq \|Vec(S1) + Vec(S4)\|$,
   if and only if $S5$ is the concatenation of $S1$ and $S4$.

6. $\|(Vec(S1) \cdot Vec(S2)) - Vec(S2)\| \simeq 0$,
   if $S2 \subseteq S1$ and the vectors include only zeros and ones

In our proposed models we only try to enforce rules 1, 3, and 4 and remove the constraint of assigning only 0 and 1 values to the vector dimensions. Our proposed models given a text create vector representations of a fixed length and assign values between 0 and 1 to the vector dimensions. We do not enforce rule 2 since we need to have a long list of biomedical synonyms and similar concepts for all documents used to train the model (In our case PUBMED titles and abstracts). We also do not enforce rules 5 and 6 because in our experiments we do not use multi-hot vector representations but assign a value between 0 and 1 to each dimension.

## B.2 Related Work

Djenouri et al. [46] introduced a cluster-based information retrieval approach that utilizes frequent and high-utility pattern mining to extract relevant patterns from a collection of objects. Two strategies, WTC and SPC, were proposed for ranking clusters based on user queries. The proposed approach was evaluated on benchmark document and tweet collections, and the results demonstrated that it improved the quality of returned objects compared to state-of-the-art information retrieval approaches, while maintaining competitive runtime performance, particularly when handling a high volume of queries.

The most similar work to our own is the work of Zamani et al. [217]. They introduced a standalone neural ranking model (SNRM) that uses an inverted index to efficiently retrieve documents from large collections. The model is optimized for information retrieval and learns a high-dimensional sparse representation for queries and documents. This representation is used to construct an inverted index, which allows the model to quickly retrieve documents at query time. The model was tested on ad-hoc retrieval tasks using newswire and web collections and was found to perform similarly to state-of-the-art neural models that rely on dense representations. The authors also demonstrated that using pseudo-relevance feedback

in the learned latent space significantly improved performance compared to competitive baselines.

Boytsov et al. [21] present a method for replacing traditional term-based retrieval with k-NN search using a non-metric, non-symmetric similarity function combining BM25 scores and IBM Model 1 log-scores. An approximate k-NN search algorithm is introduced that is significantly faster than an exact brute-force k-NN search, while still maintaining high accuracy. The proposed method is tested on a retrieval pipeline using the Stack Overflow collection and found to be both faster and more accurate than a term-based Lucene pipeline in some cases. This is the first successful application of a generic k-NN search algorithm to a combination of BM25 and IBM Model 1, and the results suggest that the method is effective for addressing semantic and syntactic mismatch in retrieval tasks. The software and derivative data used in the study are available online.

## B.3 Methods

### B.3.1 Semantic Indexing for Sentence Retrieval (SEMISER)

SEMISER is described in detail in Section 4.6.1 as it has been trained and used for retrieval in the BIOASQ competition. The rest of the models in this chapter have been evaluated on artificial data only.

### B.3.2 Sentence Embedding Model for Indexing and Retrieval using Recurrent Neural Networks (SEMIR-RNN)

The input of our model is a sentence and an n-gram while the output is a probability estimate of whether the n-gram could or could not be found in the sentence. For each token, we used a GloVe or Word2Vec pre-trained vector. When we met an out-of-vocabulary word we used the 'UNKN' token and a random embedding was assigned to it.

The structure of our model can be seen in Figure B.2. We used a bidirectional GRU [163, 166] to encode the word embeddings of the sentence. An attention mechanism (see Section B.3.4) was applied across the biGRU states creating a sentence embedding (in the simplest case the state of the last timestep was used). We used the same process using a different biGRU for the n-gram creating an n-gram embedding. The concatenation of the n-gram and sentence embeddings is then passed through an MLP which decides whether the n-gram can be found inside the sentence or not. To simplify the model and minimize the computations we also used the cosine similarity of the embeddings instead of the MLP.

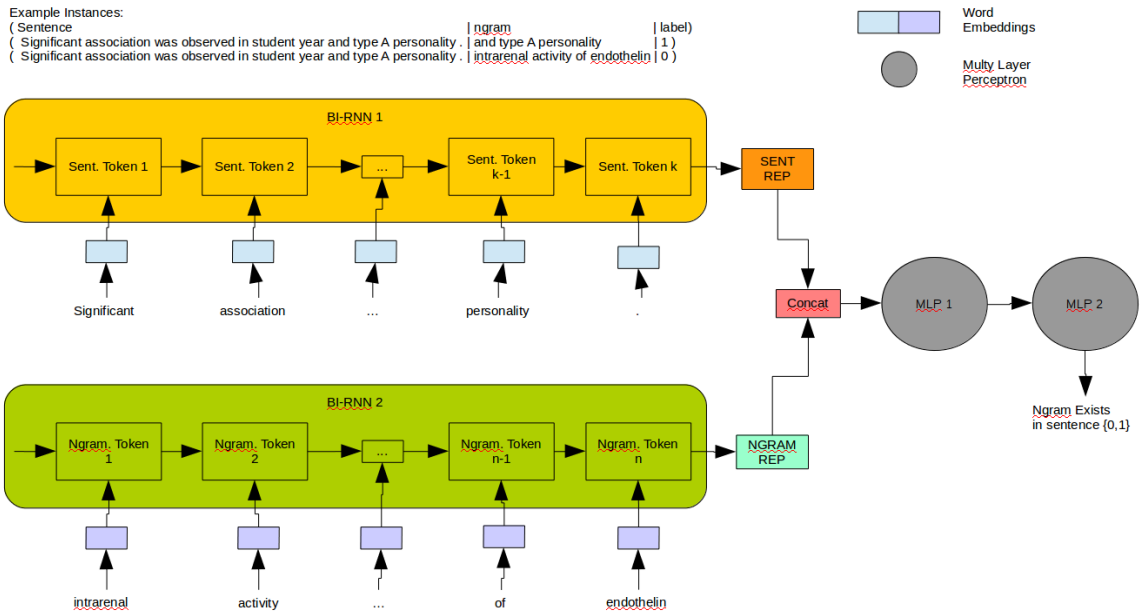


Fig. B.2 SEMIR Model using RNN and MLP.

### B.3.3 Sentence Embedding Model for Indexing and Retrieval using Convolutional Neural Networks (SEMIR-CNN)

We also experimented with convolutional layers [93] instead of RNNs. The structure of our model can be seen in figure B.3. Given the token embeddings of the sentence, we apply multiple bigram and trigram filters and then apply an attention mechanism (see Section B.3.4) creating a sentence embedding. In the same way, using different filters we created an n-gram embedding. Just like in the former model we apply an MLP (or a cosine similarity function) to the concatenation of the two embeddings.

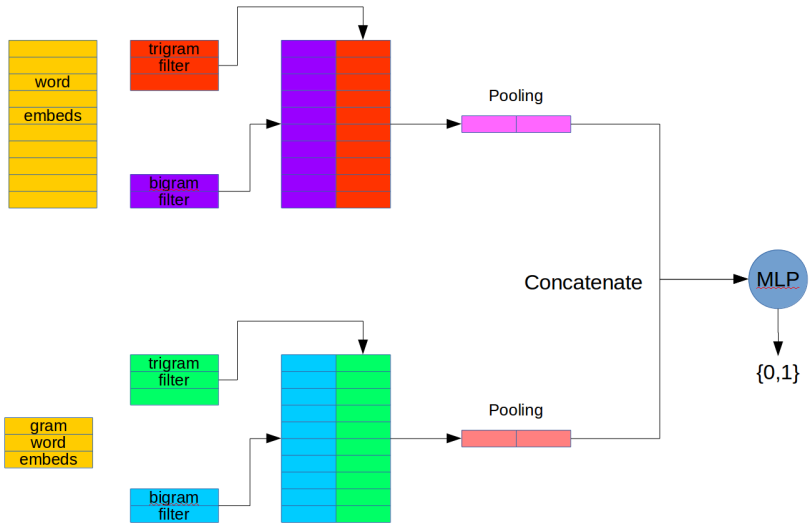


Fig. B.3 SEMIR Model using CNN and MLP.

### B.3.4 Attention on SEMIR

We experimented with several attention mechanisms that we apply to the states of the biGRU and the feature maps of the convolutions. For all k vectors in a set of vectors named V we apply:

- Last state (biGRU only),
- Min Pooling over each dimension,

$$representation = \left\langle \min_{\forall v \in V_{1..k}} v_0, ..., \min_{\forall v \in V_{1..k}} v_n \right\rangle$$

- Max Pooling over each dimension,

$$representation = \left\langle \max_{\forall v \in V_{1..k}} v_0, ..., \max_{\forall v \in V_{1..k}} v_n \right\rangle$$

- Average Pooling over each dimension,

$$representation = \left\langle \frac{\sum_{\forall v \in V_{1..k}} v_0}{k}, ..., \frac{\sum_{\forall v \in V_{1..k}} v_n}{k} \right\rangle$$

- IDF Pooling.

$$representation = \left\langle \frac{\sum_{\forall v \in V_{j=1..k}} v_0 * IDF_j}{k}, ..., \frac{\sum_{\forall v \in V_{j=1..k}} v_n * IDF_j}{k} \right\rangle$$

When applying IDF pooling we use pre-computed IDF scores as weights to multiply with the contextual embeddings extracted from the biGRU and the convolutions.

## B.4 Experiments, Results & Analysis

### B.4.1 Datasets

We apply our models described in Section B.3 to two datasets. The first dataset is the SQUAD Dataset (see Section 3.3). We detected sentences inside the paragraphs that contain the answer to the SQUAD questions and transformed the dataset so that it would be suitable for our experiments i.e., every sentence containing an answer was labeled as relevant. We also used the data of the BIOASQ competition (see Section 3.2) as an attempt to apply our models to the biomedical domain.

As SEMIR models aim to create a new unsupervised way of indexing sentences, we had to use sentences that were not included in the two datasets, to avoid fitting our models specifically to these datasets and tasks. For the BIOASQ auxiliary task we randomly selected $100k$ sentences from abstracts of the MEDLINE database, that were not included in the BioASQ dataset. For the SQUAD auxiliary task, we had to crawl and clean all of the Wikipedia articles and selected $100k$ random sentences from these articles. Once again we made sure that the randomly selected sentences were not included in the SQUAD dataset.

### B.4.2 Baselines

We compare our results to three baselines pre-trained on the same data used to train our models:

- The first baseline is a biGRU Language Model (biGRU LM). Instead of trying to predict the next token of the input using negative sampling, we exploit the fact that we use pre-trained word embeddings so in each timestep the biGRU LM has to emit an embedding which is close to the embedding of the next token. The loss function that we try to minimize is the average L2 loss of all embeddings. In order to create the sentence embedding we feed the token embeddings in the biGRU and extract the concatenation of the last timesteps of the forward GRU and the backward GRU as the sentence representation.

- The second baseline we apply is the SENT2VEC model [136]. SENT2VEC tries to create word embeddings in such a way that when removing a token from the sentence, the average of the remaining tokens' embeddings can predict the missing word. SENT2VEC's main disadvantage is the need of many examples for each word in the vocabulary. When the vocabulary is huge then an abundance of data is needed in order to train the embeddings. Nevertheless, the fact that SENT2VEC learns the embeddings of the words from scratch and trains many more parameters, makes it suitable for a strong baseline. In that case, the average of the token embeddings is used as a sentence embedding.

- The third baseline we apply is using the inverted document frequency (IDF) of each word of the sentence to create embeddings. This approach resembles a multi-hot vector but instead of using zeros and ones we use the IDF score of the word if the word is present inside the sentence. For each token, we computed its IDF in the entirety of the MEDLINE dataset. In the most memory-consuming approach where we keep the entire vocabulary, a vector of size V would be created (where V is the size of the vocabulary). In our experiments, we set an embedding size of $E$ which is much smaller than the vocabulary size. This approach would only take into account the first $E$ words of the vocabulary therefore we used a hash function that creates $E$ buckets. For each token, we computed a hash key and added the word's IDF to the corresponding bucket. Therefore, using a sentence we create an embedding that includes in dimension $i$ the sum of the IDF scores of the tokens with a hash key value $i$. In all of our experiments with our models and the baselines, we set the same embedding size ($E$ is the same embedding size in all experiments) so that the approaches are comparable.

In our models, we operate on pre-trained word embeddings that were trained on billions of training instances but we keep them frozen during the training of our models. If a word that was not encountered in the training set is found in the test set, we can use its pre-trained word embedding. If a token could not be found in the pre-trained embeddings a random one was assigned. To be fair comparing against sent2vec, when training on the auxiliary task, we selected only sentences that contained the same vocabulary as BioASQ and SQuAD so that sent2vec would have no unknown tokens in the context to better train its embeddings. We made sure that the auxiliary task and the actual sentence retrieval tasks had no sentences in common, but that sentences shared the same vocabulary.

Given a question and a set of sentences, we compute the cosine similarity between the question's embedding and each sentence embedding. We use a ReLU[1] function to extract a value between zero and one from the cosine similarity score. This score is handled as a probability score for the sentence to be relevant to the question. We use these scores and measure the Area Under the ROC Curve (AUC [23]) score using the Scikit Learn toolkit [147].

### B.4.3 Settings

We test our models in three settings:

1. Measuring AUC across the sentences of all the paragraphs which are annotated as relevant to the question. Each time we apply the Softmax function on the emissions of the model, separately for each paragraph.

[1] `https://en.wikipedia.org/wiki/Rectifier_(neural_networks)`

2. Measuring AUC across the sentences of the paragraph where the answer is present (this only applies to SQuAD). Each time we apply the Softmax function to the emissions of the model.

3. Measuring AUC across the sentences of all the paragraphs relevant to the question. Each time we apply the Softmax function on the emissions of the model. This approach is different than the first approach where the Softmax function is applied across all the sentences of each particular paragraph.

### B.4.4 Results

Several experiments where conducted and the results can be seen in Table B.1 for the SQuAD dataset and Table B.2 for the BioASQ dataset.

We experimented with several architectures of the models (using different distance measures, different pooling mechanisms, biGRU or CNN). The model yielding the best development results for the SQuAD dataset computes the question embedding and the sentences embeddings using the max of each dimension across all the emissions of the bidirectional GRU (biGRU + Max pooling) that encodes the sentence of the auxiliary task. We also had to use residuals (i.e. in each timestep of the biGRU we add the input embedding to the output of the biGRU) and compute the cosine similarity as the output layer. We barely outperform the Sent2Vec model for the first setting and fail to surpass it in setting 3. However, as we mentioned above, sent2vec trains all the embeddings of the words therefore it can train many more parameters. Additionally, we had to specifically select the data of the auxiliary task so that the auxiliary and the actual task shared the same vocabulary.

Once again we experimented with several architectures of the models for the BioASQ dataset as well. The model yielding the best development results for the BioASQ dataset computes the question embedding and the sentences embeddings using the average of each dimension across all the emissions of the bigram and trigram convolutions (CNN + Average Pooling). We also had to use residuals (i.e. in each timestep of the biGRU we add the input embedding to the output of the biGRU) and compute the cosine similarity as the output layer. We outperform all baselines by a wide margin for all three settings in the BioASQ dataset.

| AUC results on SQuAD sentence retrieval | | | |
|---|---|---|---|
| Model | setting 1 | setting 2 | setting 3 |
| IDF embeddings | 0.5009 | N/A | 0.6246 |
| biGRU LM | 0.4664 | N/A | 0.6238 |
| Sent2Vec | 0.7367 | N/A | **0.7088** |
| biGRU + max + residuals + cosine | **0.7374** | N/A | 0.6997 |

Table B.1 AUC results on the SQuAD dataset. In this general domain dataset, we observe that our model competes in equal terms to the Sent2Vec model.

| AUC results on BioASQ sentence retrieval | | | |
|---|---|---|---|
| Model | setting 1 | setting 2 | setting 3 |
| IDF embeddings | 0.4848 | 0.5857 | 0.5819 |
| biGRU LM | 0.5356 | 0.5947 | 0.5882 |
| Sent2Vec | 0.584 | 0.6091 | 0.5993 |
| CNN + average + residuals + cosine | **0.7111** | **0.6434** | **0.6185** |

Table B.2 AUC results on the BIOASQ dataset. Our model seems to surpass all other methods by a wide margin, especially for the first setting.

## B.5 Summary of Contributions

In this chapter, we present a preliminary work on dense retrieval for sentences. We evaluated new deep learning models for sentence embeddings for the task of dense retrieval. The models were trained specifically for the task of dense retrieval, which involves finding the most relevant passages in a large document for a given query. The models were trained using a combination of supervised and unsupervised learning techniques, and their performance was evaluated on two widely-used datasets, SQUAD and BIOASQ. We compared our models against three strong baselines. In our comparison against the strong baselines, our models consistently outperformed all of the other methods, demonstrating their effectiveness in the task of dense retrieval.

Unfortunately, when tested on batches of BIOASQ 7 against deep learning models for information retrieval that use attention, our dense retrieval models were outperformed by the competition and our JPDRMM model (see also Section 4.5.1). In future work, one could xplore the use of transformer-based deep neural networks for dense retrieval, with the goal of further improving the performance of these models.