Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Zelalem Gero Date

Machine learning Methods for Biomedical Keyphrase Extraction

By

Zelalem Gero
Doctor of Philosophy

Computer Science and Informatics

Joyce C. Ho, Ph.D.
Advisor

Abeed Sarker, Ph.D.
Committee Member

Imon Banerjee, Ph.D.
Committee Member

Tristan Naumann, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, PhD, MPH
Dean of the James T. Laney School of Graduate Studies

Date

Machine learning Methods for Biomedical Keyphrase Extraction

By

Zelalem Gero
B.A., Jimma University, Ethiopia, 2009
M.Sc., AAU, Ethiopia, 2012
M.Sc., Emory University, GA, 2019

Advisor: Joyce C. Ho, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2021

Abstract

Machine learning Methods for Biomedical Keyphrase Extraction
By Zelalem Gero

Due to the increased generation and digitization of text documents on the Internet and digital libraries, automated methods that can improve search, discovery and mining of the vast body of literature are essential. Efficient automated methods that extract keywords to retrieve the salient concepts of a document are shown to be of a paramount importance in text analysis, document summarization, topic detection, and recommendation systems among others.

Various machine learning approaches have been proposed to solve the problem of keyword extraction but the results still lag other tasks such as document classification. The task of keyword extraction in biomedical domain is even more daunting since the literature is highly domain specific and general methods do not translate well. To deal with these problems we propose 1) an unsupervised extraction method based on phrase-embeddings and modified pagerank algorithm which converges faster and performs better than related baseline methods; 2) A deep learning method that pays more attention to words that are central to the document's semantics; 3) a semi-supervised deep learning approach to harness vastly available unannotated biomedical data that improves keyword extraction based on uncertainty estimation. 4) An encoder-decoder based extraction for Medical Subject Heading (MeSH) indexing.

Machine learning Methods for Biomedical Keyphrase Extraction

By

Zelalem Gero
B.A., Jimma University, Ethiopia, 2009
M.Sc., AAU, Ethiopia, 2012
M.Sc., Emory University, GA, 2019

Advisor: Joyce C. Ho, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2021

Acknowledgments

I am forever indebted to all the people who have supported me throughout my graduate school journey in various capacities. Prof. Joyce Ho deserves the lion's share of all the credit for her relentless support and guidance. She has been there for me in the most difficult of times encouraging and being the voice of hope. Working with Joyce has taught me more than just research skills. It made me become a better person and I am profoundly grateful for all her help. Thank you!

I would like to thank all the members of my dissertation committee: Prof Abeed Sarker, Prof. Imon Banerjee and Dr. Tristan Naumann who have helped me with constructive feedback to improve the quality of my work.

Finally, I would like to thank the Department of Computer Science and Maths where the people made me feel at home from day one.

Thank you!!

Contents

1 Introduction **1**

1.1 What Constitutes a Keyphrase? 3

1.2 Contributions 4

1.3 Outline 6

2 Unsupervised Keyphrase Extraction **7**

2.1 Introduction 7

2.1.1 Related Work 8

2.1.2 Graph-based Methods 9

2.2 Proposed Model: NamedKeys 12

2.2.1 Candidate Keyphrase Generation 13

2.2.2 Phrase Embedding: PMCVec 14

2.2.3 Phrase Quality 16

2.2.4 Candidate Clustering and Ranking 17

2.3 Experiments 19

2.3.1 Dataset 19

2.3.2 Baseline Methods 20

2.3.3 Conclusion 25

3 Supervised Keyphrase Extraction **26**

3.1 Introduction 26

3.2 Related Work 26
3.3 Methodology 27
3.3.1 Word Embedding Layer 28
3.3.2 BiLSTM Layer 28
3.3.3 Centrality Weighting Layer 28
3.3.4 Conditional Random Fields (CRF) 30
3.4 Experiments 30
3.4.1 Datasets 30
3.4.2 Experiment Settings 31
3.4.3 Results 31
3.4.4 Conclusion 32

4 Semi-supervised Keyphrase Extraction 33
4.1 Introduction 33
4.2 Related Work 34
4.3 Methodology 34
4.3.1 BiLSTM-CRF Architecture 34
4.3.2 Self-training and Uncertainty Estimation 36
4.4 Experiments 39
4.4.1 Datasets 39
4.4.2 Experiment Settings 39
4.4.3 Evaluation Results 39
4.5 Conclusion 41

5 Mesh Indexing: Keyphrase Extraction from Controlled Vocabulary 42
5.1 Introduction 42
5.2 Related Work 44
5.3 Proposed Model: Encoder-Decoder with RL for MeSH Indexing . . . 46

5.3.1 Encoder 48
5.3.2 Decoder 49
5.3.3 Reinforcement learning for seq2seq training 50
5.4 Experimental Results 51
5.4.1 Dataset 51
5.4.2 Evaluation and Results 51
5.5 Conclusion 55

6 Conclusion and Future work **56**

Bibliography **59**

List of Figures

1.1 The conceptual framework of this dissertation. 4

2.1 The NamedKeys model pipeline. 13

2.2 A comparison of the F1 scores across the various keyphrase extraction models using exact match. 22

2.3 A comparison of the F1 scores across the various keyphrase extraction models using partial match 23

3.1 Our model architecture with the various layers. 29

4.1 A common baseline BiLSTM-CRF architecture for keyphrase extraction. 35

4.2 A self-training model architecture 36

5.1 A sample abstract with manually annotated MeSH terms. 43

5.2 The distribution of MeSH labels. 46

5.3 The Encoder-Decoder architecture with Attention 47

5.4 Examples of MeSH sequences. The left side are two cases where our model does well while the right side illustrates the challenging examples. 54

List of Tables

2.1 Comparison of the baseline models and our model using MeSH terms as golden annotations 25
2.2 The effect of various modules of NamedKeys. 25

3.1 Comparison of model performance on different datasets. 32

4.1 Comparison of model performance by fine-tuning pre-trained models. 40
4.2 Comparison of common unsupervised models and our model on PubMed dataset. 40

5.1 Average MiF scores for the baselines and our model. 52
5.2 Average MiF scores for our model and AttentionMesh using bins. . . 53
5.3 Bootstrap sample statistic using 1000 Monte Carlo simulations. . . . 53

List of Algorithms

1 Pseudo-code for iterative self-training 38

2 Pseudo-code for the REINFORCE algorithm applied to encoder-decoder 50

Chapter 1

Introduction

There has been an exponential growth in biomedical literature with over 28 million articles indexed by PubMed[1]. This growth enables researchers and practitioners to work with more literature but also presents a challenge in filtering out relevant content in reasonable time. Thus, information extraction is a key component in automated text processing as it facilitates the acquisition of structured information. Keyphrase extraction, the identification of single-world or multi-word linguistic units that concisely represent a document, is a crucial aspect of information extraction. Keyphrases help readers rapidly understand, organize, access, and share information of a document by providing a short summary of the document. Extracting keyphrases from documents is of paramount importance for natural language processing (NLP) tasks such as text summarization [7, 71], text classification [19], topic detection [43, 83], recommendation systems [59, 75], citation summarization [65] and information visualization [17]. Scientific publishers use keyphrases to identify potential reviewers for submitted articles, recommend articles to readers, and suggest missing citations to authors [4].

A variety of models have been introduced for keyphrase extraction due to its widespread use [32]. Existing keyphrase extraction systems are either supervised

[1]https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

or unsupervised. Supervised methods train classifiers on labeled examples and require large domain-specific annotations. Unfortunately, such labeled data is typically unavailable in the biomedical domain as the annotation process is labor intensive and usually necessitates significant domain expertise. Unsupervised methods, on the other hand, rely on word co-occurrence statistics from large external corpora such as Wikipedia and WordNet. Large external corpora are good statistical approximations for general domain keyphrase extraction but lack good representation in domain specific settings like biomedical text where the vocabulary can be significantly different. Moreover, many of the unsupervised approaches focus on word level co-occurrence and prefer keyphrases containing highly ranked words. These biases results towards keyphrases with more number of words. Recent unsupervised approaches such as graph-based methods and topic-based methods offer better keyphrase generation yet can suffer from a lack of diversity of the extracted keyphrases or generate phrases that may not be meaningful. Despite these efforts, the task remains challenging and the performance of current systems remains poor in comparison to other NLP tasks [49].

The challenge of keyphrase extraction is even more daunting in the biomedical domain where the text contains highly domain-specific terminologies. Given the vast amount of biomedical literature generated and digitized every year, there is a growing need to develop methods for discovering, accessing, and sharing knowledge from medical literature [66]. Significant portions of research articles published in medical journals do not have author-assigned keyphrases. Even when they come with keyphrases, the number of author-assigned keyphrases available with the articles is too limited to represent the topical content of the articles. This makes an automatic keyphrase extraction process highly desirable. Despite this need, keyphrase extraction for biomedical text has been largely ignored by the research community. Only two works [47, 70] have been introduced and either have been demonstrated on a

small set of documents or necessitated a hand-curated list of keyphrases.

A similar line of work related to keyphrase extraction is the use of Medical Subject Headings (MeSH) by a prominent medical literature database, MEDLINE. MeSH is a controlled set of terms manually assigned by human indexers in the National Library of Medicine. Even though MeSH terms make it easier to search for a document and cluster similar documents, generating MeSH terms for every document is expensive and time-consuming; new articles are not immediately indexed until 2 or 3 months later and approximately ten dollars per article is spent for the manual indexing [54]. MeSH terms can be used implicitly for automatic query expansion or explicitly in PubMed searches [51]. Compared with the commonly used keyword-based PubMed searches, MeSH indexing allows for semantic searching and searching against concepts not necessarily present in the PubMed abstract. In this dissertation, we treat MeSH indexing as a special case of keyphrase extraction where the keys are extracted from a controlled set.

1.1 What Constitutes a Keyphrase?

The goal in keyphrase extraction is to extract the most important key elements. However, what constitutes a key element is not a universally agreed upon concept. The following properties are usually thought to encompass keyphrases:

- Well-formed: Keyphrases should be well-formed words or phrases which are linguistically meaningful.
- Representative: Extracted keyphrases should reflect the major aspects discussed in the document.
- Impartiality: keyphrases should be as objective as possible which reflect the informational content of the document without personal sentiment.

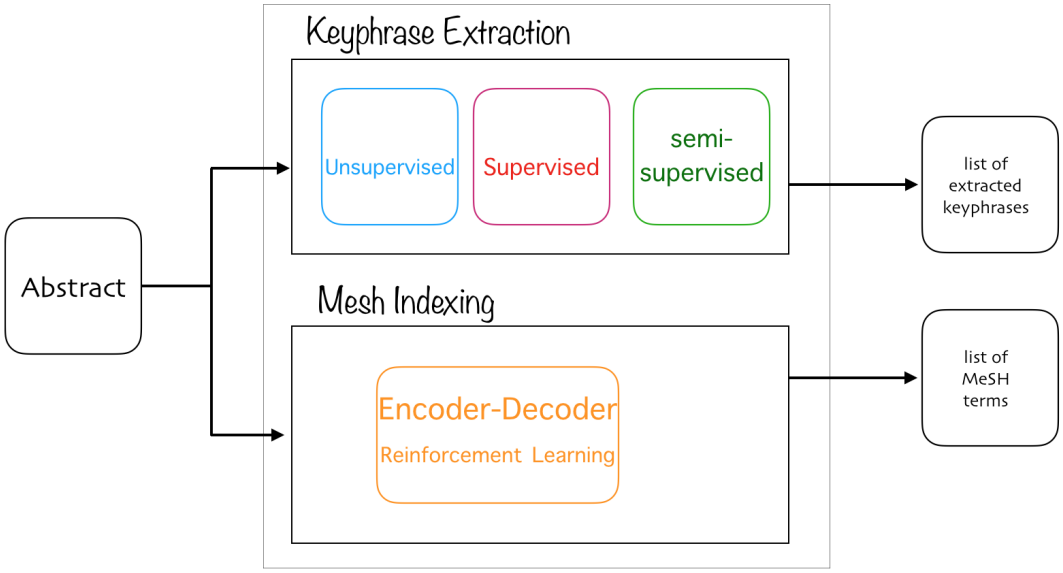


Figure 1.1: The conceptual framework of this dissertation.

- Specific: Keyphrases should be as specific as possible by selecting keys which are characteristic of a document to differentiate it from other similar documents.
- Minimal: The extracted keys should be different from each other.
- Exhaustive: The extracted keyphrases should cover all the subjects covered in the document.

1.2 Contributions

Despite the enormous growth of machine learning and NLP techniques in the last decade, extraction of semantically meaningful and coherent keyphrases from text is still a non-trivial task. In this dissertation, we develop several reliable keyphrase extraction models focused on biomedical documents by solving the the two aforementioned problems: keyphrase extraction and MeSH indexing. The overall conceptual framework of our work is shown in Figure 1.1.

Our contributions are summarized as follows:

- Unsupervised extraction models: Since most of the existing unsupervised models work better in the general domain, we propose a method specifically focused on the biomedical domain as the literature is domain-specific. First, we develop a biomedical embedding model which learns vector representations for words and phrases [25]. As biomedical literature contains multi-word phrases, phrase embeddings capture the semantics much better. Secondly, we develop a keyphrase extraction model that utilizes the trained phrase embeddings and a modified PageRank [61] algorithms to extract words and phrases which are syntactically and semantically meaningful [26].

- Supervised extraction model: Unsupervised methods work well when there is not enough annotated training data is available. However, in most settings, their performance is not on par with supervised methods. The problem of keyphrase extraction can be modeled as a sequence tagging task where each token is classified as either being part of the keyphrase or not. This setting uses the Bidirectional Long Short Memory Network-Conditional Random Fields architecture commonly applied for such tasks as Named Entity Recognition, yet the extracted keyphrases may not capture the main gist of the document. We improve on this architecture by introducing the idea of word centrality using attention since keyphrases are central to the document [28].

- Semi-supervised extraction model: To take advantage of the large amount of unlabelled data, we propose a semi-supervised method that starts with a small amount of labeled data to gradually use more unlabeled data for better performance. Unlike previous works which fail to account for uncertainty of the psuedo-labels, we use Monte-Carlo simulation for uncertainty estimation [27]. Our model then selects a subset of psuedo-labeled datapoints from the large unlabeled dataset to re-train the model iteratively.

- MeSH indexing using seq2seq: Tagging biomedical documents with list of MeSH terms is a closly related task with keyphrase extraction. We consider the controlled MeSH vocabulary as the super-set of keyphrases from which we extract based on the input text. We develop a sequence to sequence model (seq2seq) which is trained using reinforcement learning with a designed reward function. This allows the model to learn the high level correlation between the MeSH labels and improve the overall performance.

1.3 Outline

The rest of this dissertation is structured in four chapters. In Chapter 2, we introduce the commonly used unsupervised keyphrase extraction methods. We detail the steps and features each method uses. Then we present our proposed unsupervised extraction model. In the Experiments section, we present comparative results. In Chapters 3 and 4, we discuss our supervised and semi-supervised extraction models respectively. Comparisons with baseline models is also presented. In the final chapter, we propose sequence-to-sequence (seq2seq) models for MeSH indexing task. We consider the task as generating MeSH labels as sequences which depend on each other. By incorporating reinforcement learning in the training of the seq2seq model, we alleviate the problem of label-invariance while generating individual labels.

Chapter 2

Unsupervised Keyphrase Extraction

2.1 Introduction

Unsupervised methods have the advantage that they do not rely on human annotated datasets to train. Hence, the focus is on the features extraction that can generalize for all datasets despite the domain. Most of the unsupervised keyphrase extraction systems share the following steps:

1. Candidate Selection: Select potential keyphrases using predefined heuristics. Such heuristics commonly remove stopwords, punctuation, and words that have some specific parts of speech tags.

2. Candidate Ranking: Candidate ranking is performed using various features such as frequency of occurrence and proximity to other candidates.

3. Post-processing: Top words from the ranked list above will be selected and if necessary joined to form longer keyphrases.

In the subsections that follow, we detail how these steps are used by each of the

methods. We also point out the limitations of the baseline models when applied to biomedical data. Then we discuss our proposed model in detail with comparative results with the baselines.

2.1.1 Related Work

Unsupervised keyphrase extraction methods can be broadly categorized into: statistical methods, graph-based methods, and topic-based methods.

Statistical Methods

Statistical methods focus on statistical features based on the current corpus that do not require external corpora. The most common features used by statistical methods are the position of the first occurrence of a candidate, word frequency, casing, and how often a candidate word appears in different sentences.

Term-frequency Inverse document frequency (Tf-Idf) is the most widely used statistical baseline. The basic premise of Tf-Idf is that terms that occur frequently in a document are likely to be more important while terms that occur across the corpus are not specific enough to be very important. Tf-Idf for a term t in document d across a corpus D is calculated as:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D),$$

where $tf(t, d)$ is the frequency of the term t in a given document d and $idf(t, D) = \log N/(Nd)$ where N is the total number of documents in the corpus and Nd is the number of documents that contain the term.

KP-Miner [21] is a keyphrase extraction system that exploits various types of statistical information beyond the Tf and Idf scores. It follows a quite effective filtering process of candidate phrases and uses a scoring function similar to Tf-Idf.

KP-Miner generates candidate tokens that are not be separated by punctuation marks or stopwords. These candidates are filtered based on the least allowable frequency in addition to the number of words after which a phrase appears for the first time. Finally, it ranks the candidate phrases considering the Tf and Idf scores as well as the term position and a boosting factor for compound terms over the single terms.

YAKE [16] preprocesses the text by splitting it into individual tokens. In addition to position and frequency of each token, YAKE uses five additional statistical metrics that capture context information and the spread of the terms into the document:

- Casing (Wcase): the casing of a word
- Word Position (WPosition): words occurring at the beginning of a document are valued more
- Word Frequency (WFreq): the observed frequency of the word
- Word Relatedness to Context (WRel): the number of different terms that occur to the left/right side of the candidate word
- Word DifSentence (WDifSentence): how often a candidate word appears within different sentences

Finally, all these features are aggregated for the computation of the $S(t)$ score for each term (the smaller the value, the more important the word t would be).

$$S(t) = \frac{WRel \cdot WPos}{Wcase + \frac{WFreq}{WRel} + \frac{WDif}{WRel}}$$

2.1.2 Graph-based Methods

The basic idea in graph-based ranking is to create a graph from a document that has as nodes the candidate phrases from the document and each edge of this graph

connects related candidate keyphrases. The final goal is the ranking of the nodes using a graph-based ranking method, such as PageRank.

TextRank [55] is the first widely adopted graph-based keyphrase extraction method. The text is tokenized and Part of Speech (POS) tagging is done for every token. Based on the POS tags, only nouns and adjectives are kept. Next the candidates are added to the graph as nodes and an edge is added between the nodes that co-occur within a window of N words. The graph is undirected and unweighted. Each node will be initialized with a score of 1, then the PageRank algorithm runs until it converges. Specifically, for a node V_i, the corresponding score function that is iteratively computed is:

$$S(V_i) = (1 - \lambda) + \lambda \sum_{j \subset N(V_i)} \frac{1}{N(V_j)} S(V_j)$$

where $N(V_{\mathrm{i}})$ is the set of neighbors of V_{i}, $N(V_{\mathrm{j}})$ is the set of neighbors of V_{j}, and λ is the probability of jumping from one node to another.

SingleRank [81] extends TextRank and incorporates weights to edges. It uses co-occurrence statistics for context information among words. Each edge weight is equal to the number of co-occurrences of the two corresponding words. Then, the score function for a node V_{i} is computed in a similar way:

$$WS(V_i) = (1 - \lambda) + \lambda \sum_{j \subset N(V_i)} \frac{CO_{ij}}{\sum_{Vk \subset N(V_j)} CO_{jk}} WS(V_j)$$

where CO_{ij} is the number of co-occurrences of word i and word j.

After convergence, for each continuous sequence of nouns and adjectives in the text document, the scores of the constituent words are summed up and the T top-ranked candidates are returned as keyphrases.

PositionRank [22] is based on word-word co-occurrences and their corresponding

position in the text. It incorporates all positions of a word into a biased weighted PageRank. The final scores from PageRank are then used to rank the candidates.

Topic-based Methods

Topic-based keyphrase extraction methods try to extract keyphrases that are representative of all the topics discussed in a text document. These methods usually apply clustering techniques or Latent Dirichlet Allocation (LDA) [9] to detect the main topics discussed.

TopicRank [15], is the widely used topic-based model. It preprocesses the text to extract the candidate phrases. Then, the candidate phrases are grouped into separate topics using hierarchical agglomerative clustering. Instead of a word graph, a graph of topics is constructed whose edges are weighted based on a measure that considers phrases' offset positions in the text. Finally, TextRank is used to rank the topics and a keyphrase candidate is selected from each of the N most important topics.

MultipartiteRank [14] is a method similar to TopicRank which introduces an in-between step where edge weights are adjusted to capture position information giving bias towards keyphrase candidates occurring earlier in the document.

Keyphrase extraction specifically for biomedical domain has been experimented by few researchers. In [47], Li et al. extract noun phrases from medical literature as keyphrase candidates and assign weights to extracted noun phrases for a medical document based on how important they are to that document and how domain-specific they are in the medical domain using WordNet lexical database and Specialist Lexicon. Even though this work is a pioneer in the extraction of keyphrases from medical documents, the use of a very small test set of 60 documents is not large enough for conclusive results. In [70], Sarkar presents a hybrid approach to keyphrase extraction from medical documents. The approach is an amalgamation of two methods: the first one assigns weights to candidate keyphrases based on combination of features such

as position, term frequency, inverse document frequency and the second one assign weights to candidate keyphrases using some knowledge about their similarities to the structure and characteristics of keyphrases available in the memory (stored list of keyphrases). This approach necessitates the availability of hand-curated keyphrases in memory to learn from, making it harder to use in an unsupervised setting.

Although the aforementioned methods perform well, they have limitations that can be improved. The methods are based on single-word candidates which make it difficult to capture keyphrases with multiple words. They attempt to remedy this by concatenating words during post-processing but our results show that such post-processing leads to semantically meaningless phrases. Most of the keyphrases that are constructed from single words during post-processing end up becoming fragments that are not syntactically valid.

2.2 Proposed Model: NamedKeys

Extracting keyphrases from text can be considered as selecting phrases that capture the gist of the document and are also semantically and syntactically correct. In [76], these two measures are referred to as informativeness and phraseness. Informativeness measures how good a phrase is in capturing the main theme of the document while phraseness measures the likelihood of a sequence of words to be a meaningful phrase. We propose `PMCVec`– a new keyphrase extraction algorithm, that produces informative and meaningful keyphrases for biomedical text. Our model, illustrated in Figure 2.1, consists of the following steps: (1) new candidate keyphrase generation mechanisms to construct an extensive keyphrase candidate set; (2) a new phrase-embedding representation for the document and the phrases to better measure the informativeness of a given phrase; (3) a new "phraseness" metric to assign a normalized score for every phrase generated from the corpus; and (4) a ranking and

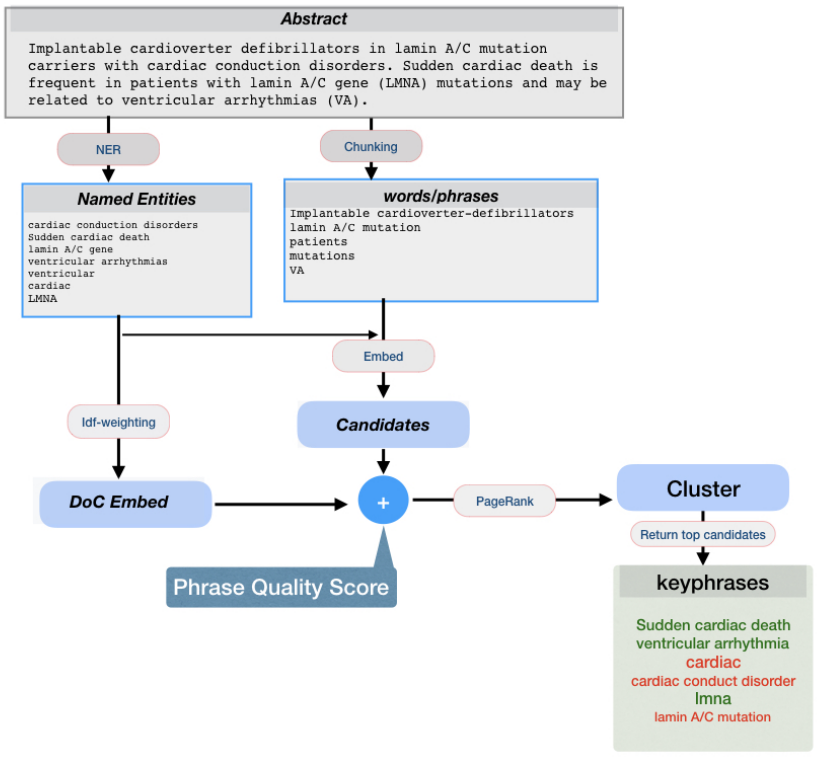


Figure 2.1: The NamedKeys model pipeline.

clustering module that ranks the candidate phrases and clusters the keyphrases to ensure that the extracted keyphrases are diverse. The details for each of the four steps are discussed in the following subsections.

2.2.1 Candidate Keyphrase Generation

We propose two new mechanisms for generating keyphrase candidates. The first process uses named entity recognition (NER) to extract information such as disease names, medication, symptoms, and chemicals. Instead of constructing multi-word phrases based on important consecutively occurring words, we start with phrases as a single unit of representation. We observed that many keyphrases in biomedical literature capture concepts such as chemicals, diseases, cell types, proteins, and gene named entities. Biomedical-specific NER has been shown to help identify problems and symptoms a patient has exhibited, tests that have been run, and treatments that

have been administered [10]. Unfortunately, biomedical named entities may not occur frequently enough in the text, and thus are often not suggested by existing keyphrase extraction tools. Thus, we used SciSpacy [60], a specialized NLP library for processing biomedical texts, to detect all the named entities in the text. SciSpacy contains different modules for chemicals and disease named entities; cell types; proteins and gene named entities; and cell lines, DNA, RNA and cancer named entities.

The second mechanism finds phrases that are not named entities but are still potentially meaningful. Rather than rely only on the NER, we propose a generic approach to extract more candidate phrases. `PMCVec` chunks the text by identifying potential keyphrase boundaries using stopwords and punctuation [70]. From the generated chunks, those which belong to the following parts of speech will be retained: 'JJ', 'JJR', 'JJS', 'NN', 'NNS', 'NNP', 'NNPS'. We perform the parts of speech selection using Genia Tagger[1], a biomedical tool for text processing. Although stop words and punctuations will never occur in any proposed keyphrase, it provides a systematic methodology for generating variable n-graph keyphrases. The detected named entities from the NER process and from this chunking process are combined to construct the candidate keyphrase pool.

2.2.2 Phrase Embedding: PMCVec

The next step of `PMCVec` focuses on identifying the candidate keyphrases with high informativeness measures. We propose the use of word embeddings to help rank the candidate phrases based on closeness to the document. Word embeddings are dense-low dimensional vector representations of words such that related words are close in vector space. Each dimension in the vector represents a feature of a word, and the vector can, in theory, capture both semantic and syntactic features of the word. Word2Vec [30], Glove [29], and FastText [12] are commonly used approaches to train

[1]http://www.nactem.ac.uk/GENIA/tagger/

word vectors. Unfortunately, many of the common word embedding approaches and pre-trained vectors focus on unigrams, while key concepts in biomedical literature are often expressed as multi-word phrases [62].

We develop a phrase-based embedding model to capture the semantic and syntactic relation between terms (or n-grams). We use a data-driven approach of extracting a commonly occurring sequence of words and learn embeddings for the extracted phrases along with the single words. [29] showed that the presence of unigram words intermixed with multi-word phrases improves the performance of embedding models. To avoid pre-specification of the number of words for a phrase, we used a similar idea as the second mechanism in the keyphrase candidate generation step. We identify potential phrase boundaries using stopwords and punctuations (excluding the hyphen). A sequence of words that occur more than a pre-defined threshold (100 is used for our experiments) are considered potential phrases. Phrases are merged into a single word (e.g., prostate cancer becomes prostate_cancer) in the order they originally appeared in the text. The multi-word phrases are trained with all the single words in the corpus. We use the Word2Vec tool to train our phrase embedding model on over 27 million PubMed abstracts. As a result, our embeddings can capture the semantic relation between related concepts like "Hypertension" and "High Blood Pressure".

Once the vector representations are available for all the terms in a document as well as the candidate keyphrases, `PMCVec` measures the informativeness of the candidates compared to the candidate vectors of the document. Given the importance of the named entities in the biomedical documents, we represent the document, d, using the IDF-weighted sum of the named entities:

$$d = \frac{\sum_{i=1}^{n} idf(i, d, D) \cdot w_i}{\sum_{i=1}^{n} idf(i, d, D)},$$

where w_i is the corresponding vector representation of the named entity and $idf(i, d, D)$ is the inverse document frequency of the named entity. This is used for attenuating the effect of terms that occur too often in the collection to be meaningful for relevance determination. Then to calculate the informativeness of a given candidate phrase, we compute the cosine similarity between the document and the keyphrase representation, w_k:

$$Similarity(d, w_k) = \frac{d^T w_k}{||d||_2 ||w_k||_2} \tag{2.1}$$

2.2.3 Phrase Quality

A candidate keyphrase can have a high cosine similarity to the document and can still not be a syntactically meaningful phrase. As an example, "ventricular arrhythmias vary" could have a high cosine similarity to a document discussing "ventricular arrhythmias" but should not be ranked high since the phrase is not syntactically sound. A better phrase would be just "ventricular arrhythmias". Although there are several common phrase ranking criteria [18, 36], we found they offered a poor trade-off between phrase frequency, constituent word frequency, and phrase length. For example, point-wise mutual information (PMI) is often used to find good collocation pairs as it calculates the probability of co-occurrence relative to the probabilities of the occurrence of each word. Conversely, phrases that contain frequently occurring words will have small PMI scores even if the phrase is good. To measure the phraseness of a candidate keyphrase, we propose "Information Frequency (Info_Freq)", a new ranking metric based on the phrase frequency and constituent words frequency in the corpus [25]. This criteria achieves the state-of-the-art performance when the resulting distributed word representations are evaluated on five benchmark datasets for biomedical semantic similarity. Our metric adds a multiplier to the PMI index

that captures the overall frequency of the phrase:

$$\text{Info\_Freq}(x, y) = \log \frac{p(x, y)}{p(x)p(y)} \cdot \log(\text{freq}(x, y)) \tag{2.2}$$

where $p(x, y)$ is the probability of the two words occurring together, $p(x)$ is the probability of the first word in the text and $p(y)$ is the probability of the second word in the text. Thus, Info_Freq measures the phraseness of a sequence of words by considering how often the phrase and the constituent words occur. Info_Freq is calculated for all the candidate keyphrases and the scores are normalized to lie between 0 and 1.

2.2.4 Candidate Clustering and Ranking

The final two steps of `PMCVec` are to rank the candidates using the informativeness and meaningfulness measures, as well as cluster the candidate keyphrases to avoid redundancy of keyphrases. While phrase embedding and phrase quality capture the general informativeness and meaningfulness of a given candidate phrase respectively, we use local co-occurrence of two candidates to capture the local relationship between the phrases within the context of the given document using a weighted PageRank algorithm [32]. The document is represented as a weighted undirected graph, where vertices correspond to the words/phrases and the edges represent the co-occurrence relations between two terms. Two vertices are connected if they occur in the same sentences. For a graph with V vertices and E edges, the score for vertex v_i is calculated as:

$$S(v_i) = (1 - \beta)W_i + \beta \sum_{j \subseteq in(v_i)} \frac{sim(v_i, v_j)}{|out(v_j)|} S(v_j), \tag{2.3}$$

where W_i is the importance weight of vertex i measured as the average of its similarity to the document vector and its phrase quality, sim is the cosine similarity between keyphrases v_i and v_j, out is the number of outgoing edges of keyphrase v_j, and β is a damping factor. Thus, high scores reflect phrases that capture the gist of the document and are also semantically and syntactically correct.

Two common problems with keyphrase extraction algorithms are overgeneration and redundancy of keyphrases. Overgeneration errors occur when a system correctly predicts a candidate as a keyphrase because it contains a word that appears frequently in the associated document, but at the same time erroneously outputs other candidates as keyphrases because they contain the same word. Redundancy errors occur when a system correctly identifies a candidate as a keyphrase, but at the same time outputs a semantically equivalent candidate (e.g., it's alias) as a keyphrase. A recent study performed error analysis on the various algorithms and showed that 52-64% of the errors were due to overgeneration and redundancy of keyphrases [32]. For existing algorithms that rely on frequency, it can be difficult to avoid overgeneration errors as rejecting a non-keyphrase containing a word with a high term frequency might negatively impact the precision of the algorithm. On the other hand, redundancy errors occur due to the inability to detect that two candidates are semantically equivalent.

To overcome the overgeneration and redundancy error, we propose clustering the candidate keyphrases based on their semantic similarity. The cluster analysis achieves two purposes: identify keyphrases that are semantically similar and diversify the generated document keyphrases. The clustering algorithm uses the cosine similarity scores for all pairs of the candidate keyphrases to identify keyphrases that are similar. The importance of the cluster is then calculated based on the average distance of each of its candidates to the document. The cluster importance weights are then normalized to sum up to 1, and are used to determine the composition of the extracted keyphrases. For example, a cluster with a weight of 0.5 will provide approximately

50% of the final generated keyphrases, while a cluster with 0.1 weight will contribute 10%. To further avoid redundancy, keyphrases will not be selected if they are too similar (e.g., $sim(v_i, v_j) \geq \alpha$ where $\alpha = 0.75$ in our experiments). While any clustering algorithm based on distances can be used, we used the Affinity Propagation clustering algorithm [23] with a damping factor of 0.85 and Euclidean affinity. One benefit for Affinity Propagation is that the number of clusters is automatically learned from the data.

2.3 Experiments

2.3.1 Dataset

Keyphrase extraction for biomedical text has not been widely studied except for the two works mentioned in the related works [47, 70]. Thus, we construct a new benchmark dataset based on the PubMed Central Open Access Subset articles. This dataset was constructed by selecting all the abstracts which contain at least 5 author-provided keyphrases. Five is chosen as the minimum number of keyphrases since most evaluation benchmarks are done at a minimum of five keyphrase extraction. Since the focus of this work is keyphrase extraction, we propose that the author-provided keyphrases serve as appropriate summarizations of their articles. Thus, we did not consider abstracts where there are no author-provided keyphrases.

While the PubMed Central Open Access Subset contains over 28 million articles at the time of download, only 3049 articles had a title, abstract, and at least five author-provided keyphrases found in the abstract. A value of 0.85 is used for the damping factor β as this gave the best score on a separate training set of 2000 abstracts. This training set is different from this benchmark test set as we used abstracts with less than five author provided keyphrases.

In our benchmark dataset, we provide the following fields for each article:

- title: the title of the article,
- abstractText:the abstract of the article,
- keyphrases: a list of keyphrases provided by the authors

2.3.2 Baseline Methods

We compare the NamedKeys model with the state-of-the-art keyphrase extraction approaches implemented by Boudin et al. [13]. Graph-based approaches commonly use PageRank algorithm to determine the importance of candidates by using incoming and outgoing vertices to/from each candidate. Candidates with connections to other important candidates will have higher rank while candidates with fewer connections or connections to less important vertices will be ranked lower. Statistical-based approaches rely on features extracted from the document such as the position of first occurrence of a candidate, word frequency, casing, and how often a candidate word appears in different sentences. These approaches commonly use external corpus like Wikipedia to construct the features and perform well in a general domain while graph-based methods have the benefit of performing well in any domain since they do not depend on specific corpus features. We could not find the implementations of two baselines [53, 82] that reported performing well in the general domain. Our implementation of their models performed worse than the other baselines used here. Hence, we did not report the scores from those baselines. For the graph-based approaches the following are used as baselines:

- **MultiPartiteRank** [14]: An approach that encodes the topical information within a multipartite graph structure and exploits their mutually reinforcing relationship to improve candidate ranking.
- **PositionRank** [22]: An algorithm that incorporates information from all positions of a word's occurrences into a biased PageRank algorithm.

- **SingleRank** [80]: A method that encodes the mutual influences of multiple documents within a cluster context.
- **TextRank** [55]: A model that accounts for the local context of a text unit (vertex) and the information recursively drawn from the entire text (graph).
- **TopicRank** [15]: A graph-based method that relies on a topical representation of the document.

For the statistical-based approaches the following are used as baselines:

- **TF-IDF**: Term frequency-inverse document frequency, a common weighting technique in information retrieval and text mining.
- **KP Miner** [21]: A model that makes use of the first position a candidate phrase appears and the TF-IDF measure as a weight.
- **YAKE** [16]: A feature-based system for multi-lingual keyword extraction from single documents.

We first evaluated the algorithms based on exact match as a function of the number of phrases extracted. Figure 2.2 shows the F1 scores for our NamedKeys model and the other baseline approaches based on the extracted number of phrases between ten and thirty in increments of five. NamedKeys consistently achieves the highest score with a performance gain of up to 35% from the next best method. We also observe that almost all of the algorithms achieve the highest F1 score when evaluated at 30. This intuitively makes sense as the algorithms can achieve higher recall without much loss in precision as the number of extracted phrases increases. From the figure, the three statistical approaches (KP Miner, YAKE and TFIDF) achieve the worst F1 scores overall. This can be attributed to the fact that such approaches mainly focus on the number of times the candidate occurs and the position of first occurrence.

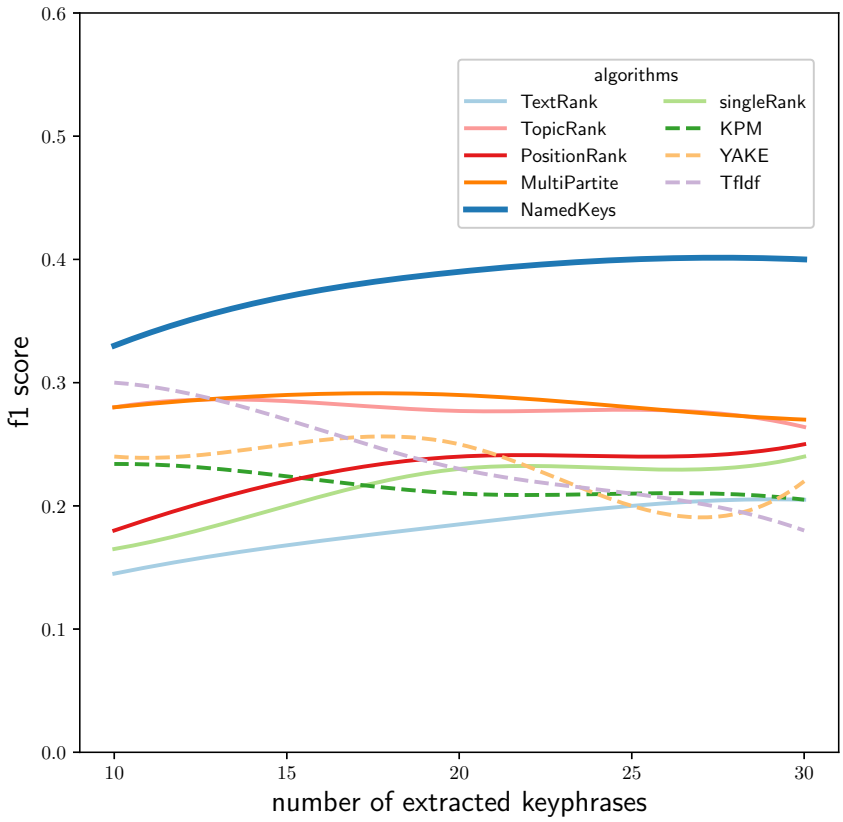


Figure 2.2: A comparison of the F1 scores across the various keyphrase extraction models using exact match.

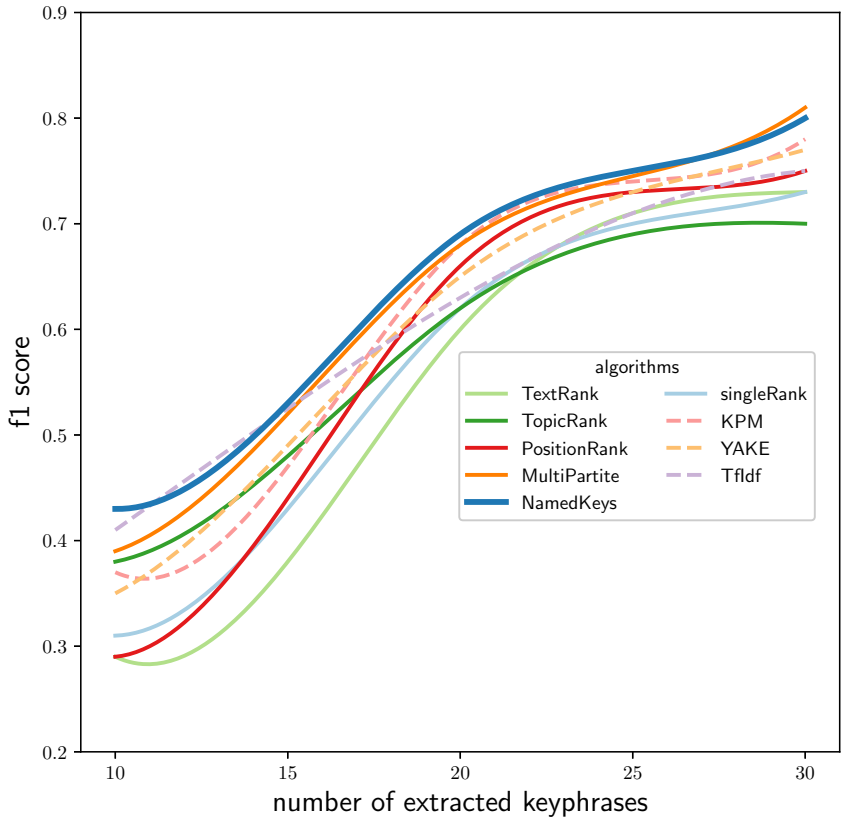


Figure 2.3: A comparison of the F1 scores across the various keyphrase extraction models using partial match

These heuristics are not typically important in biomedical documents as phrases can be very important without having to occur multiple times.

Exact match evaluation serves as a lower bound on the model performance as partial matches are considered incorrect. For example, if the keyphrase is "high blood pressure", a model that identifies "blood pressure" will obtain the same score as another model that fails to identify "blood pressure". An alternative performance measure is the Message Understanding Conference (MUC) standard of partial evaluation. According to the MUC, the following terminologies are defined to compare the system extracted keyphrases against the gold annotations.

- **Correct (COR)**: Outputs are the same
- **Incorrect (INC)**: Outputs don't match
- **Partial (PAR)**: Outputs match partially
- **Missing (MIS)**: Golden annotations not captured by the extraction system.
- **Spurious (SPU)**: System extracts keys which are not in the golden annotations.

The partial match scoring is then calculated as:

$$Possible = COR + INC + PAR + MIS \tag{2.4}$$

$$Actual = COR + INC + PAR + SPU \tag{2.5}$$

$$Precision = \frac{COR + 0.5 \cdot PAR}{Actual} \tag{2.6}$$

$$Recall = \frac{COR + 0.5 \cdot PAR}{Possible} \tag{2.7}$$

In Figure 2.3, we show the performance comparison using MUC partial evaluation. When comparing with Figure 2.2 illustrates that the partial match evaluation results are much better than the exact match across all the baselines. The score variance among the methods is also much less. We also observe that NamedKeys consistently outperforms the other models except at the tail ends (low number of extracted keyphrases and high number of keyphrases).

Due to high variability of author provided keyphrases, we experimented with using Medical Subject Headings (MeSH) as the golden annotations. Since MeSH terms are curated list of terms, only a few are usually assigned as keywords by authors. We used abstracts which contain at least 1 MeSH term to compare the extractive results. As shown in 2.1, the performance degrades across all the extraction algorithms.

Table 2.1: Comparison of the baseline models and our model using MeSH terms as golden annotations

	SingleRank	PositionRank	TopicRank	NamedKeys
F1@5	0.11	0.12	0.13	0.13
F1@10	0.23	0.25	0.24	0.24
F1@15	0.29	0.31	0.29	0.31

We also performed an ablation experiment to quantify the performance gains for the various modules in our model as shown in Table 2.2.

Method		**@10**	**@15**	**@20**	**@25**	**@30**
No NER	**P**	0.24	0.20	0.17	0.15	0.13
	R	0.17	0.20	0.24	0.26	0.30
	F1	0.19	0.20	0.20	0.19	0.18
NER + Embed	**P**	0.39	0.37	0.37	0.36	0.36
	R	0.22	0.29	0.36	0.39	0.42
	F1	0.28	0.32	0.36	0.37	0.39
NER + Embed+ chunk	**P**	**0.39**	**0.37**	**0.38**	**0.37**	**0.37**
	R	0.22	0.29	0.36	0.38	0.42
	F1	0.28	0.32	0.37	0.37	0.39
NER + Embed + chunk + phraseQuality	**P**	0.38	0.35	0.33	0.31	0.31
	R	**0.30**	**0.40**	**0.48**	**0.55**	**0.55**
	F1	**0.34**	**0.37**	**0.39**	**0.40**	**0.40**

Table 2.2: The effect of various modules of NamedKeys.

2.3.3 Conclusion

In this chapter, we introduced NamedKeys – a method composed of four modules: NER, phrase embedding, phrase quality score and similarity-based clustering for keyphrase extraction from biomedical documents. To evaluate the proposed method, we created a new publicly available benchmark dataset from PubMed Central Open Access articles. Our unsupervised approach results in performances better than existing baselines at various numbers of keyphrases extracted.

Chapter 3

Supervised Keyphrase Extraction

3.1 Introduction

Supervised learning is a powerful machine learning method when labeled data is available. Many text processing tasks such as text classification, sentiment analysis, and information extraction gain performance improvements with supervised methods even in the presence of a modest amount of labeled data. In this chapter, we propose a new supervised deep learning model for the task of keyphrase extraction from biomedical literature. Our model performance is compared with several common baselines.

3.2 Related Work

Supervised keyphrase extraction methods often treat the problem as a binary classification task [1, 78, 79], where learning algorithms such as support vector machines [37, 85] and maximum entropy [42, 87] are used. Other supervised approaches pose the problem as a ranking between candidates [85]. The candidates keys are extracted using statistical (e.g., tf-idf, number of occurrences, first occurrence of the key) and structural features (e.g., part of speech tags). Conditional Random Fields (CRFs) have also been explored on top of the statistical and structural features in [44].

Deep learning based models have also been used for supervised keyphrase extraction. Word embeddings are used to measure the relatedness between words in graph-based models [82]. [89] used a Recurrent Neural Network (RNN) based approach to identify keyphrases in Twitter data. The model addresses the problem as sequence labeling for very short text, where a joint-layer RNN is used to capture the semantic dependencies in the input sequence. In [69], the authors used an attention-based neural network to extract keyphrases from scientific documents by retrieving additional information from other sentences within the same document. Sahrawat et. al [68] evaluate the effect of various pretrained word embeddings in extracting keyphrases from benchmark datasets. Al-Zaidy et.al. [1] employ a Long Short-Term Memory (LSTM) with a CRF layer to model keyphrase extraction as a sequence labelling task. Even though the latest deep learning models have shown much improvement over traditional methods, they fail to capture the centrality of the keyphrases which represents a salient feature of the document. To this end, we propose a centrality layer on top of a Bidirectional LSTM (BiLSTM) to constrain the importance of each token with regard to the document. This ensures tokens chosen as keyphrases are the ones representing the main topics of the document.

3.3 Methodology

The keyphrase extraction task is formulated as a sequence labelling task. Given a document $X = w_1, w_2, \cdots, w_t$ where w_i is the i^{th} word and t is the number of words in the document, we predict the labels $y = y_1, y_2, \cdots, y_t$ where each label y_i is whether word w_i is a keyphrase or not.

3.3.1 Word Embedding Layer

Each word in the document is represented by pre-trained low-dimensional vector representations. Any pre-trained vector representation can be used, and we experiment with various pre-trained embeddings such as Glove [64], word2vec [56], BERT [20] and BioBERT [46]. The impact of each embedding type is evaluated and discussed in the experiments section.

3.3.2 BiLSTM Layer

This layer is used to encode each document to obtain the local contextual representation. A forward and backward LSTM are used to read the input sequence from left to right, $\overrightarrow{h_1}, \overrightarrow{h_2}, \cdots, \overrightarrow{h_t}$, and right to left, $\overleftarrow{h_1}, \overleftarrow{h_2}, \cdots, \overleftarrow{h_t}$, respectively. The outputs from the two directions are concatenated and summed for the final hidden state representation of the document, $H = [\sum_{i=1}^{t} \overrightarrow{h_i}, \sum_{i=1}^{t} \overleftarrow{h_i}]$.

3.3.3 Centrality Weighting Layer

Sequence labelling is commonly used for other token encoding tasks such as Named Entity Recognition (NER) where the task is to determine whether a token is a named entity or not. However, keyphrase extraction is different from other sequence labelling tasks (for example NER) in that, we want the tokens to capture the main gist of the document. This is in contrast to NER where the importance of the token is irrelevant as long as it is a named entity. To incorporate the idea of centrality, we learn the similarity between each token and the document embedding, H to bias towards tokens which are central to the document.

For words $\{w_1, w_2, \cdots, w_t\}$ in a document D, we learn the centrality weight for each word $\alpha_1, \alpha_2, \cdots, \alpha_t$. The output representation, z_i for each word is then the centrality weight multiplied by the output of the BiLSTM, $z_i = [\alpha_i \overrightarrow{h_i}, \alpha_i \overleftarrow{h_i}]$

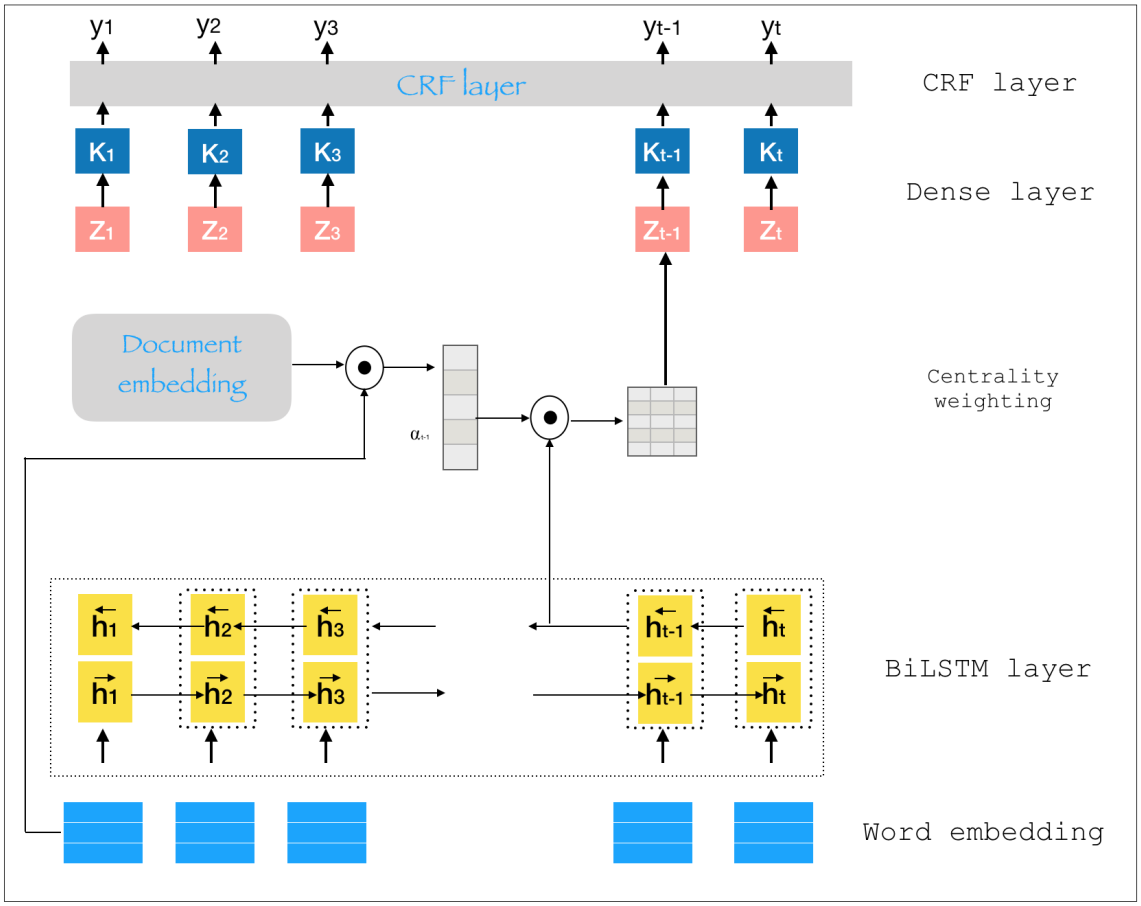


Figure 3.1: Our model architecture with the various layers.

3.3.4 Conditional Random Fields (CRF)

CRFs are widely used to model sequence labeling tasks [44]. Given the input document as sequence of tokens, CRF produces a probability distribution over the output label sequence using the dependencies among the labels of the entire input sequence. This formulation considers the correlations between neighboring labels and allows joint decoding for the best sequence of labels for the input sequence, rather than decoding each label independently. Figure 3.1 illustrates our model architecture with the various layers.

3.4 Experiments

3.4.1 Datasets

We ran our experiment on 2 publicly available keyphrase datasets: PubMed [26], INSPEC [35]. PubMed consists of 2532 articles from PubMed Central Open Access Subset with at least 5 author-provided keyphrases while INSPEC contains 200 abstracts of scientific journal papers from Computer Science collected between the years 1998 and 2002. Each document in INSPEC has two sets of keywords assigned: the controlled keywords, which are manually controlled assigned keywords that appear in the INSPEC thesaurus but may not appear in the document, and the uncontrolled keywords which are freely assigned by the editors. The union of both sets is considered as the ground-truth in this work. Since we use a sequence labeling formulation of the keyphrase extraction problem, the abstract/keyphrases data pairs are prepared such that each document is a sequence of word tokens, each with a positive label if it occurs in a keyphrase, or with a negative label.

3.4.2 Experiment Settings

As baseline models, we train BiLSTM, BiLSTM-CRF, and DAKE[69] with different embedding vectors. The word embeddings are initialized with 100-dimension Glove pre-trained embedding vectors [29], 768-dimension BERT embeddings [20], and 768-dimension BioBERT embeddings [46].

The BiLSTM, and BiLSTM-CRF are optimized during training using stochastic gradient descent with learning rate 0.0001. Gradient clipping of 5.0 is used to prevent the gradient from overflows during back-propagation. In addition, we use dropout to avoid over-fitting. We select the model with the best F1 score on the validation set on three runs.

3.4.3 Results

The BiLSTM and BiLSTM-CRF baselines are trained using Glove and Word2Vec pretrained embeddings. The results in Table 3.1 are using Glove embeddings which perform slightly better than Word2Vec embeddings. For the BERT baseline, we used BERT embeddings with BiLSTM-CRF setting. Similarly, we used the BioBERT embeddings with BiLSTM-CRF for the BioBERT baseline. DAKE is another state-of-the- art baseline which uses sentence enriching process from all the documents using sentence embedding. To replicate their work, we used the Bert model to extract sentence embeddings for each document and enrich the representation. Finally, our model is trained using BERT word embeddings for the INSPEC dataset and BioBERT embeddings for the PubMed dataset. In addition to the word embeddings, we have the centrality constraint layer using the embedding vectors.

The performance comparison of the baselines and our model is shown in Table 3.1. Our model performs much better on the PubMed dataset compared to the baselines. The improvement gained from our model is not as large on the INSPEC dataset. We hypothesize that for the centrality constraint to be effective, the input sequence

Table 3.1: Comparison of model performance on different datasets.

Model	PubMed	INSPEC
BiLSTM	0.543	0.427
BiLSTM-CRF	0.554	0.453
BERT	0.604	0.581
BioBERT	0.622	0.464
DAKE	0.623	0.463
Ours	**0.644**	**0.586**

should be relatively longer. The sentences in the INSPEC dataset are much shorter hence the difficulty in learning the central theme.

3.4.4 Conclusion

In this chapter, we proposed a keyphrase extraction method that focuses on identifying words which are central to the document semantics. The problem of keyphrase extraction is posed as a sequence labeling task where each token is tagged as either a keyphrase or not. In addition to our novel centrality constraint layer, we have used BiLSTM layers to capture the long term dependencies among the input sequences. Finally, we have a CRF layer which is well suited to capture the dependencies from the output labels. Empirical results on two datasets shows that our method gains significant improvement in the PubMed dataset while performing slightly better on the INSPEC dataset.

Chapter 4

Semi-supervised Keyphrase Extraction

4.1 Introduction

As we have seen in the previous chapter, with the availability of labeled data, supervised methods show performance gains in comparison to unsupervised methods. However, in many domains, the availability of manually labeled data is limited. In contrast, large amount of unlabeled text data is being generated and digitized every moment. This opens up an opportunity to use these large unlabeled datasets to train machine learning models. In this chapter, we detail how we combine small amount of labeled data with large unlabeled data to develop a semi-supervised machine learning model that performs better than the supervised counterparts. We also detail how Monte-Carlo dropout can be used to approximate model uncertainty to select samples from the unlabeled data.

4.2 Related Work

Since deep learning models require significant labeled data, self-training for keyphrase extraction has been explored recently [45, 91]. Conceptually, the model is first trained on the labeled data and then used to generate pseudo-labels for the unlabeled data. A subset of the pseudo-labeled data is then used to re-train the model and this process is iteratively done until all the unlabeled data have been used. Even though their approaches show performance gains over baseline models, the uncertainty of the model is not incorporated which can lead to poor learning and noise propagation. We propose to incorporate the uncertainty of the psuedo-label during self-training for further improvements.

4.3 Methodology

The keyphrase extraction task is formulated as a sequence labelling task. Given a document $X = w_1, w_2, \cdots, w_t$ where w_i is the i^{th} token and t is the number of tokens in the document, we predict the labels $Y = \{k_B, k_I, k_O\}$ where k_B, k_I and k_O denote whether the token is the beginning of, part of, or not a part of a keyphrase, respectively. The baseline deep learning model we employ is the commonly used BiLSTM-CRF architecture [1, 45, 68, 91] shown in Figure 4.1. We first briefly describe the BiLSTM-CRF model before introducing self-training and uncertainty estimation for keyphrase extraction.

4.3.1 BiLSTM-CRF Architecture

Token Emebedding

Each token, w_i, is represented by a low-dimensional vector representations x_i. Any pre-trained word embedding can be used such as Glove [64], word2vec [56], SciBERT

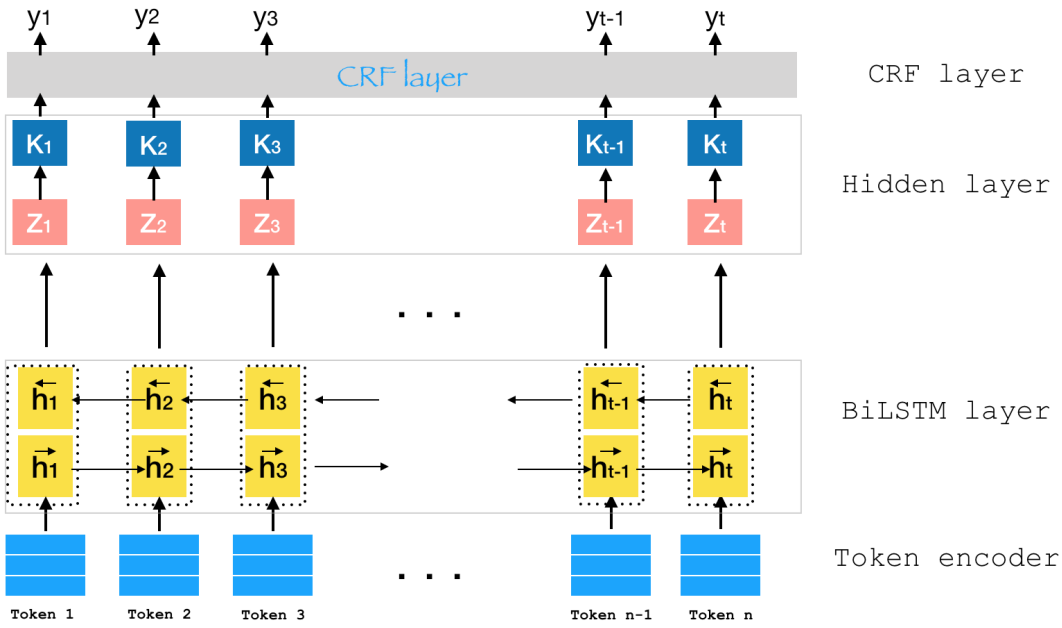


Figure 4.1: A common baseline BiLSTM-CRF architecture for keyphrase extraction.

[6] and BioBERT [46]. Contextualized embeddings such as SciBERT have been shown to provide better results [68].

BiLSTM Layer

A BiLSTM layer is used to encode each document into a local contextual representation. The BiLSTM generates two feature representations, $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$, for each x_i using a forward and backward LSTM, respectively. The two representations are concatenated and then passed to an affine transformation $k_t = W_a \overleftrightarrow{[\overrightarrow{h_t}; \overleftarrow{h_t}]}$.

CRF

Given the sequence of tokens, CRF produces a probability distribution over the output label sequence using the dependencies among the labels of the entire input sequence [44]. Given a transition matrix Γ where $\Gamma_{i,j}$ is the transition score from class y_{t-1} to y_t, the score of an output label sequence s is given by $s(s, y) = \sum_{t=1}^{n} \Gamma_{y_{t-1}, y_t} + K_t, y_t$. The overall likelihood score for a given sequence is then calculated by exponentiating

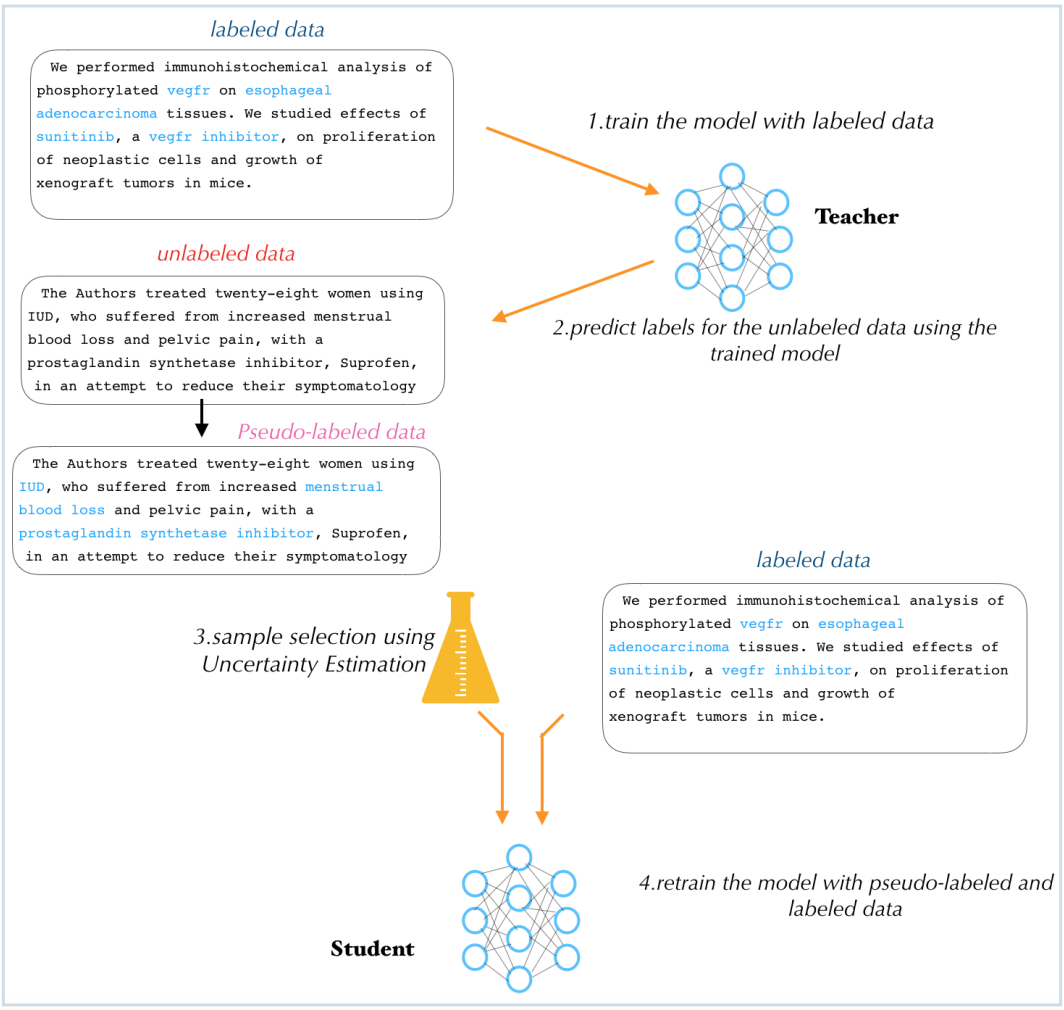


Figure 4.2: A self-training model architecture

the individual scores and normalizing over all possible output sequences.

4.3.2 Self-training and Uncertainty Estimation

Self-training is a semi-supervised approach which has state-of-the art performances across several applications [48, 67, 74]. Under the self-training paradigm, a teacher model is trained on a small amount of labeled data ($\mathcal{D}_l$) and used to generate pseudo-labels on unlabeled data ($\mathcal{D}_u$). A subset of the pseudo-labeled data is then combined with the labeled data to train a second model called a student model. The student then becomes the teacher and this process is repeated until convergence is achieved.

The overall process of the self-training architecture is shown in Figure 4.2.

While several self-training keyphrases extraction models have been proposed [45, 91], they fail to consider the teacher uncertainty. These implementations only sample pseudo-labeled instances where the model confidence is high in a single pass. Predictive probabilities from a softmax output are erroneously taken as model confidence. Gal et al. [24] demonstrate that a model can be uncertain in its predictions even with a high softmax output. This can lead to poor learning and noise propagation through self-training on wrong pseudo-labels [58]. Moreover, selecting samples where the model is very confident may not improve the performance of the student model as these may already be correctly classified. However, selecting samples where the model is least confident can make it difficult to learn anything important. Mukherjee et al. [58] proposed to select examples based on the uncertainty of the teacher model to improve the self-training process by modeling a distribution over the parameters through Bayesian Neural Networks to reflect model uncertainty. Unfortunately, direct adoption of this framework is not straightforward as questions arise from the multiple pseudo-keyphrase annotations associated with each document.

Based on the promising results of using uncertainty to improve the self-training process, we introduce a new uncertainty-based self-training model for keyphrase extraction. For each sample in the unlabeled data ($\mathcal{D}_u$), we use Monte-Carlo dropout [24] to simulate a Bayesian approximation to quantify the uncertainty associated with the teacher model f_t with corresponding model parameters W. This means that M forward passes are performed where stochastic dropouts are applied to each hidden layer ($\tilde{W}_m$) to approximate the model output as a random sample from the posterior distribution as in [58]. It is important to note that this process will create different pseudo keyphrases for each document since dropouts are activated during inference as well. Thus for each unlabeled sample x_u, there are M pseudo-labels for each token in the document, $y_1^*, \cdots, y_M^*$. The pseudo-labels are used to compute the stochastic

mean and variance of x_u:

$$E(y) = \frac{1}{M}\sum_{m=1}^{M} y_m^*(x) \tag{4.1}$$

$$Var(y) \approx \frac{1}{M}\sum_{m=1}^{M} y_m^*(x)^\top y_m^*(x) - E(y)^\top E(y) \tag{4.2}$$

From these, model uncertainty is approximated by the summary of variance of the model outputs from the multiple forward passes. The uncertainty for a given unlabeled document is the mean of the uncertainties of the individual tokens. This gives us the pseudo-labels with their corresponding uncertainties $u_1, u_2, ..., u_m$ for each unlabeled document x_u. Pseudo-labeled samples with low uncertainty values are considered easier while high uncertainty valued samples are harder for the teacher model to predict. To enhance the student learning, we select samples with the average uncertainty value less than a threshold (we used 0.2 since this gives the best results on the validation set). This helps with selecting some samples where the teacher model is not very certain. Algorithm 1 outlines our uncertainty-based self-training process for keyphrase extraction.

Algorithm 1: Pseudo-code for iterative self-training

```
Train f_t teacher model with parameters W on D_l;
while not converged do
    for x ∈ D_u do
        for m ∈ {1, ··· , M} do
            W̃_m ∼ Dropout(W) ;
            y*_m = softmax(f^(W_m)(x)) ;
        end
        Calculate stochastic mean and variance of x ;
    end
    Sample instances with uncertainty less than a given threshold (α) ;
    Retrain model W using the combined data ;
end
```

4.4 Experiments

4.4.1 Datasets

We ran our experiment on a publicly available scientific keyphrase dataset: PubMed [26]. PubMed contains 2532 articles from PubMed Central Open Access Subset with at least 5 author-provided keyphrases. Since we use a sequence labeling formulation, the document/keyphrases data pairs are prepared such that each document is a sequence of word tokens, where the positive labels (k_B, k_I) are used if the word occurs in a keyphrase and a negative label (k_O) if it is not part of the keyphrase.

For the self-training based model, we use an unlabeled dataset, PubMed-Medline[1] which contains over 28 million abstracts of biomedical journals.

4.4.2 Experiment Settings

We split the dataset into 80%, 10%, and 10% for training, validation and testing, respectively. Our model is compared against two BiLSTM-CRF with two different word embeddings: 768-dimension SciBERT [6], and 768-dimension BioBERT [46]. The BiLSTM-CRF models are optimized during training using stochastic gradient descent with a learning rate 0.0001. Gradient clipping of 5.0 is used to prevent the gradient from overflows during back-propagation. In addition, we use dropout to avoid over-fitting. We evaluate the models using the F1 score on the test set using three different runs.

4.4.3 Evaluation Results

To quantify the performance benefits of self-training for keyphrase extraction, we have used two of the best performing pre-trained models commonly used: SciBERT and BioBERT. These pre-trained models already achieve state-of-the art performances in

[1] `https://www.nlm.nih.gov/databases/download/pubmed_medline.html`

many downstream tasks. We fine-tuned the pre-trained models by adding a BiLSTM and CRF layers with small labeled data available. After fine-tuning on the small labeled data, we use the self-training module to keep sampling from the unlabeled set.

The performance comparison of the baselines and our model is shown in Table 4.1. Since pre-trained models already use large amount of unlabeled data, it's usually cumbersome to squeeze out performance improvements. Our model shows significant performance gain on the dataset compared to the baselines.

For comparison with common unsupervised approaches, we ranked our keyphrase tagging based on the predicted model uncertainty. In Table 4.2, we show F1 scores when extracting 5,10, and 15 keyphrases from a document. The results show that unsupervised methods lag significantly behind their semi-supervised counterpart as our model performs better.

Table 4.1: Comparison of model performance by fine-tuning pre-trained models.

Model	F1 score
BiLSTM-CRF (SciBERT)	0.765 ($\pm$0.003)
BiLSTM-CRF (BioBERT)	0.768 ($\pm$0.003)
SciBERT + JLSD [45]	0.765 ($\pm$0.003)
SciBERT_sahrawat [68]	0.766 ($\pm$0.002)
Ours (BioBERT + CRF + Self)	0.773 ($\pm$0.002)

Table 4.2: Comparison of common unsupervised models and our model on PubMed dataset.

	SingleRank	PositionRank	TopicRank	Ours
F1@5	15.2	18.3	26.4	**36.2**
F1@10	16.3	18.3	28.7	**54.3**
F1@15	19.2	20.9	29.2	**64.5**

4.5 Conclusion

In this chapter, we proposed a new uncertainty-based self-training keyphrase extraction method that utilizes unlabeled data to augment small labeled training data. We proposed the use of Monte-Carlo dropout to approximate the model uncertainty for each pseudo-labeled document. The uncertainty is then used to sample specific documents to retrain the model using the combined data. This iterative Teacher-Student model training is performed until convergence is achieved. The empirical results on the PubMed dataset showcase that self-training can provide an performance improvement, especially when there is a significant unlabeled corpus.

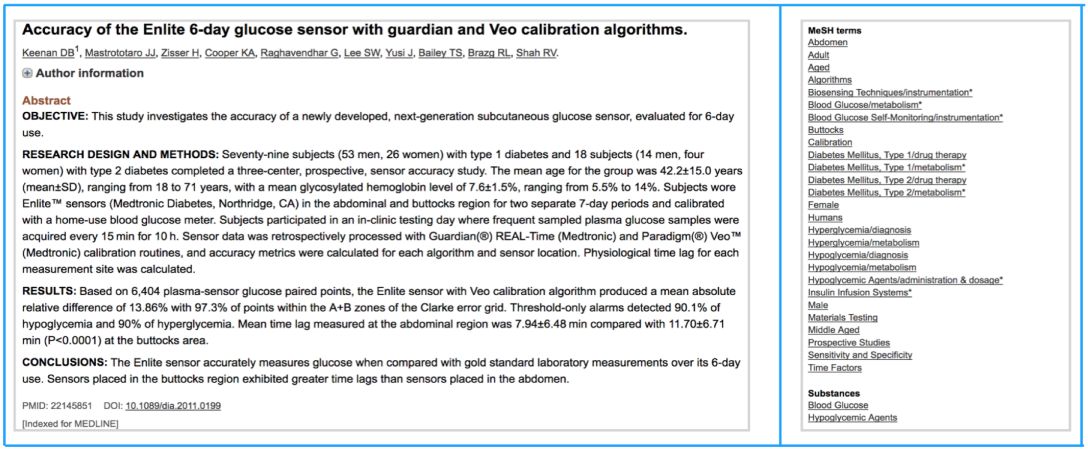
Chapter 5

Mesh Indexing: Keyphrase Extraction from Controlled Vocabulary

5.1 Introduction

MEDLINE is the U.S. National Library of Medicine® (NLM) premier bibliographic database that contains more than 30 million references to journal articles in life sciences with a concentration on biomedicine. A distinctive feature of MEDLINE is that the records are indexed with NLM Medical Subject Headings (MeSH®)[1]. MeSH is a comprehensive controlled vocabulary, which has been developed and maintained by National Library of Medicine. There are over 29 million MeSH main headings. MeSH indexing is the task of assigning a set of hierarchically-organized terminology to citations. An example of manually indexed MEDLINE abstract with the MeSH terms is shown in Figure 5.1. MeSH terms can then be used implicitly for automatic query expansion or explicitly in PubMed searches [51]. Compared with the commonly used

[1]https://www.nlm.nih.gov/bsd/medline.html



Accuracy of the Enlite 6-day glucose sensor with guardian and Veo calibration algorithms.

Keenan DB[1], Mastrototaro JJ, Zisser H, Cooper KA, Raghavendhar G, Lee SW, Yusi J, Bailey TS, Brazg RL, Shah RV.

Author information

Abstract

OBJECTIVE: This study investigates the accuracy of a newly developed, next-generation subcutaneous glucose sensor, evaluated for 6-day use.

RESEARCH DESIGN AND METHODS: Seventy-nine subjects (53 men, 26 women) with type 1 diabetes and 18 subjects (14 men, four women) with type 2 diabetes completed a three-center, prospective, sensor accuracy study. The mean age for the group was 42.2±15.0 years (mean±SD), ranging from 18 to 71 years, with a mean glycosylated hemoglobin level of 7.6±1.5%, ranging from 5.5% to 14%. Subjects wore Enlite™ sensors (Medtronic Diabetes, Northridge, CA) in the abdominal and buttocks region for two separate 7-day periods and calibrated with a home-use blood glucose meter. Subjects participated in an in-clinic testing day where frequent sampled plasma glucose samples were acquired every 15 min for 10 h. Sensor data was retrospectively processed with Guardian(®) REAL-Time (Medtronic) and Paradigm(®) Veo™ (Medtronic) calibration routines, and accuracy metrics were calculated for each algorithm and sensor location. Physiological time lag for each measurement site was calculated.

RESULTS: Based on 6,404 plasma-sensor glucose paired points, the Enlite sensor with Veo calibration algorithm produced a mean absolute relative difference of 13.86% with 97.3% of points within the A+B zones of the Clarke error grid. Threshold-only alarms detected 90.1% of hypoglycemia and 90% of hyperglycemia. Mean time lag measured at the abdominal region was 7.94±6.48 min compared with 11.70±6.71 min (P<0.0001) at the buttocks area.

CONCLUSIONS: The Enlite sensor accurately measures glucose when compared with gold standard laboratory measurements over its 6-day use. Sensors placed in the buttocks region exhibited greater time lags than sensors placed in the abdomen.

PMID: 22145851 DOI: 10.1089/dia.2011.0199

[Indexed for MEDLINE]

MeSH terms

Abdomen
Adult
Aged
Algorithms
Biosensing Techniques/instrumentation*
Blood Glucose/metabolism*
Blood Glucose Self-Monitoring/instrumentation*
Buttocks
Calibration
Diabetes Mellitus, Type 1/drug therapy
Diabetes Mellitus, Type 1/metabolism*
Diabetes Mellitus, Type 2/drug therapy
Diabetes Mellitus, Type 2/metabolism*
Female
Humans
Hyperglycemia/diagnosis
Hyperglycemia/metabolism
Hypoglycemia/diagnosis
Hypoglycemia/metabolism
Hypoglycemic Agents/administration & dosage*
Insulin Infusion Systems*
Male
Materials Testing
Middle Aged
Prospective Studies
Sensitivity and Specificity
Time Factors

Substances

Blood Glucose
Hypoglycemic Agents

Figure 5.1: A sample abstract with manually annotated MeSH terms.

keyword-based PubMed searches, MeSH indexing allows for semantic searching and searching against concepts not necessarily present in the PubMed abstract. MeSH has also been used in many other applications in biomedical text mining, such as document summarization [8], document clustering [31, 34], word sense disambiguation [40], and question answering [90]. Thus, accurate MeSH indexing of biomedical documents is crucial for the biomedical researchers in formulating novel scientific hypothesis and discovering new knowledge.

To maintain the high quality of term assignment to citations, NLM carefully indexes citations with MeSH terms manually. On average, each citation is indexed by 13 Mesh terms. Even though the manual indexing is very valuable, it is taxing to the curators and expensive. It is estimated that $9.4 is spent to annotate a single MEDLINE citation [57]. Moreover, the size of the biomedical literature is growing exponentially over the past few years [50]. This makes the task of indexing manually even more difficult and prohibitively expensive. To alleviate this, NLM uses MTI (Medical Text Indexer), a software system that suggests suitable MeSH terms to human curators [3]. However, the curators still have to read the entire document and assign MeSH terms manually. The MTI output is merely used as a method to narrow the large selection space to a reasonable number for the human curators.

Manually indexing MEDLINE documents is not only labor intensive and expensive, but also time consuming. Hence, new documents are not indexed until 2 or 3 months after publication [54]. Given the enormity of the task, various machine learning approaches have been proposed to solve the MeSH indexing problem. In the sections that follow, we discuss the related works and detail our proposed method which poses the task as sequence-to-sequence problem using an Encoder-Decoder Transformer architecture.

5.2 Related Work

MeSH indexing can be approached as a multi-label text classification (MLC) task with each MeSH heading as a class. In this setup, each citation will have multiple headings assigned as class. Various techniques that use the MLC method have been applied to solve the problem of MeSH indexing; Naïve Bayes [38], support vector machines [39], K-nearest neighbors [77], and Learning to Rank [33]. Most recent methods use the combination of two or more of the above methods. MTI which is developed by NLM to provide MeSH recommendations to human indexers has a module which generates candidate MeSH terms and another to filter and rank the candidates. It uses MetaMap [2], another text processing tool developed by NLM which identifies medical concepts from text and maps to UMLS (Unified Medical Language System) [11] concepts. UMLS is a set of files and software that brings together many health and biomedical vocabularies and standards to enable interoperability between computer systems. MTI summarizes text using MetaMap and MeSH to recommend terms. Using nearest neighbors, MTI extracts more MeSH from related citations and ranks the final recommendations.

Due to their recent advances and performance gains in many tasks, various deep learning based methods have been applied to solve the MeSH indexing problem.

The first approach, DeepMesh [63] like MTI uses two modules: the first to generate MeSH candidates and predict the number of output MeSH terms, and the second is to rank the candidates and take the highest-ranked predicted number of MeSH terms as output. DeepMeSH uses TF-IDF and document to vector (D2V) schemes to represent each abstract and generate MeSH candidates using binary classifiers and a k-nearest neighbor (KNN) method solver using these features. TF-IDF is a traditional weighted bag of word sparse representation of the text and D2V learns a deep semantic representation of the text.

MeshNow [54] reformulates the MeSH indexing task as a ranking problem and applies a learning to rank method. MeshNow is a three-step process: First, given a target article, it obtains an initial list of candidate MeSH terms from three unique sources (KNN, MLC, and MTI recommendation). Next, it applies a learning-to-rank algorithm to sort the candidate MeSH terms based on the learned associations between the document text and each candidate MeSH term. Finally, it prunes the ranked list and returns a number of top candidates as the final system output.

Recently Qin et al [41] developed an interpretable MeSH indexer using deep learning and the attention mechanism. Starting with an input abstract, title and journal name, words in the document are embedded and fed to BiGRU (BiDirectional Gated Recurrent Unit) to derive context-aware representations. KNN-derived articles from training corpus are identified and frequent MeSH terms in them are included as candidate annotations for the document. MeSH terms are embedded, and only those candidates are further considered using the attention mechanism. The attention mechanism assigns attention weights to each word with respect to each candidate MeSH term, which leads to a MeSH-specific document representation. Finally, they use MeSH-specific document representations as input to perform the multi-label classifications. For each candidate MeSH term of a document, the model outputs a probability. Even though this model uses attention mechanism to boost performance, it

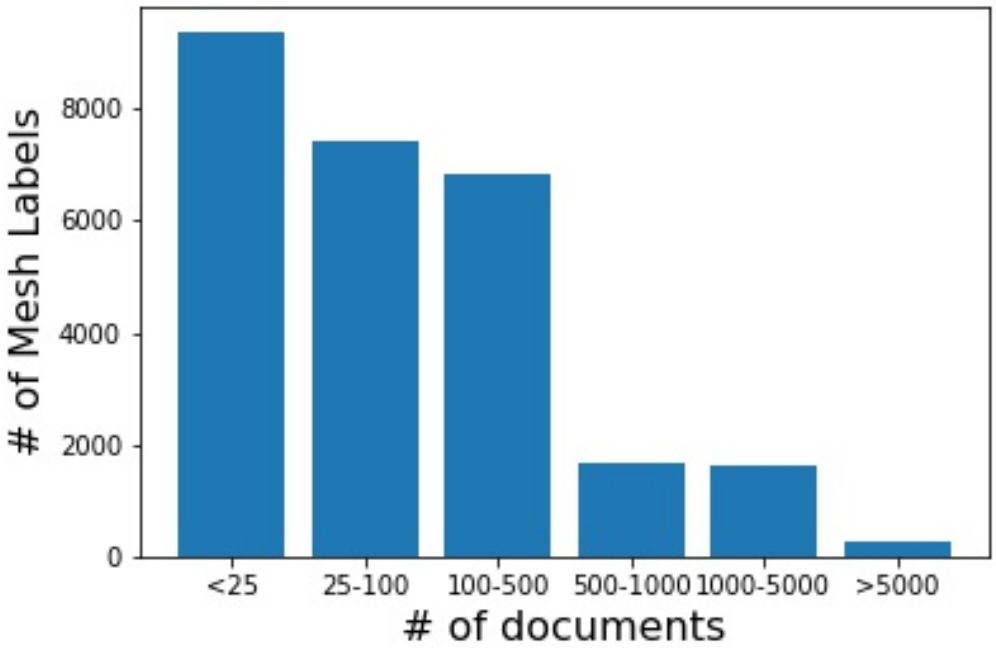


Figure 5.2: The distribution of MeSH labels.

is a multi-label classification model and hence the tail-end labels suffer from lack of enough training data.

5.3 Proposed Model: Encoder-Decoder with RL for MeSH Indexing

MeSH indexing can be considered as a multi-label classification (MLC) task. In an MLC setting, each label is trained independently of the other labels. In the case of MeSH indexing, there is a complex dependency between each of the assigned labels. The existence of one label can hint to the inclusion/exclusion of another label. Moreover, there is a large skewness between the MeSH label distribution as shown in Figure 5.2. Most of the MeSH labels have less than 100 supporting documents. Training separate classifiers using such few data points is a challenge. To capture the high-order

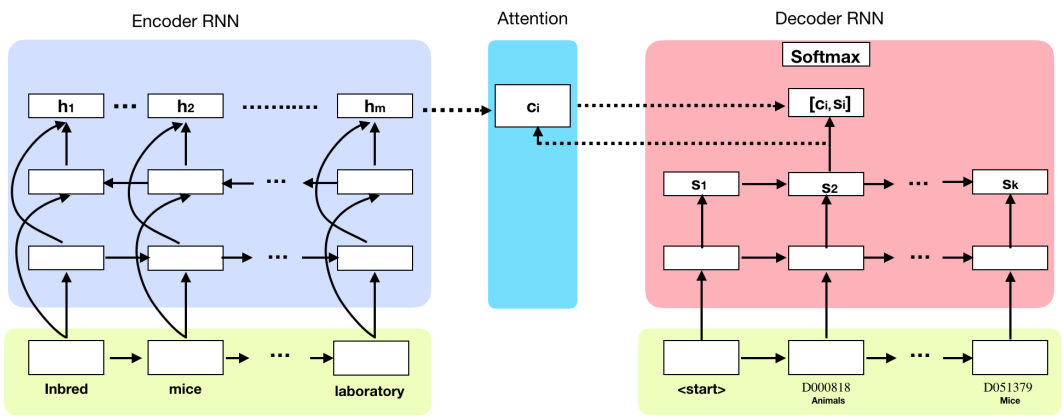


Figure 5.3: The Encoder-Decoder architecture with Attention

correlation between labels, we propose a a method based on the sequence-to-sequence architecture (seq2seq) [73]. Seq2seq models are shown to effectively capture dependencies between labels for tasks ranging from Machine Translation [5], to Question Answering [86] and DNA function prediction [88].

Seq2seq models' performance have been impressive in tasks where the input and output sequences are relatively short. When dealing with long sequences, vanilla seq2seq models have difficulty coping with long-range dependencies. The RNN building blocks employed by seq2seq models struggle to encode long sentences. For the task we are dealing with, MeSH indexing, input sequences can be as long as thousands of tokens making it difficult to capture the entire input context. To deal with such issues, the attention mechanism was proposed and has been shown to be effective [52].

We employ the Encoder-Decoder with attention, shown in Figure 5.3, to model the MeSH indexing task as a seq2seq problem. NLP tasks which employ the Encoder-Decoder architecture assume a (loose) correspondence between input tokens and output tokens. In such tasks, the length of the input and output sequences are usually comparable. In contrast, for MeSH indexing, the input sequence is significantly longer than the output sequence. There is no one-to-one token correspondence as is common

in other tasks such as machine translation which poses challenges during encoding. Another challenge is the ordering of output sequences. MeSH labels are a set of unordered terms where there is no inherent natural order. In machine translation, for example, both the input language as well as the output language have a syntax and semantic rules which govern the order of the sequences. To solve this, we used the MeSH terms' distribution frequency as an ordering criteria. Even though attention-based Encoder-Decoder models have shown to produce competitive results, two common problems still persist. The first is exposure bias where the model struggles when ground truth tokens are not fed to the decoder during inference. The second problem is the inconsistency between training and test objectives. During training, the log-loss objective is used for learning. However, in inference time we want to maximize an F1-score. Since F1-score is not differential, it can not be directly maximized as a model objective. To deal with these issues, we propose using a reinforcement algorithm on top of the Encoder-Decoder model. The main components of the our architecture are summarized below.

5.3.1 Encoder

The encoder reads the input sequence and encodes to a fixed-length internal representation. This is implemented as a BiLSTM. For each input word, the forward and backward hidden states are computed using the LSTM passes and are concatenated for the final representation of the word.

$$\overrightarrow{\mathrm{h_i}} = \overrightarrow{\mathrm{LSTM}}(\overrightarrow{\mathrm{h_{i-1}}}, (w_i)) \tag{5.1}$$

$$\overleftarrow{\mathrm{h_i}} = \overleftarrow{\mathrm{LSTM}}(\overleftarrow{\mathrm{h_{i-1}}}, (w_i)) \tag{5.2}$$

$$h_i = [\overrightarrow{\mathrm{h_i}}, \overleftarrow{\mathrm{h_i}}] \tag{5.3}$$

5.3.2 Decoder

Based on the input received from the encoder and its current state, the decoder generates a label and also updates its own state for the next time step until the end-of-sequence token is generated.

$$s_t = LSTM(s_{t-1}, [y_{t-1}; c_t]), \tag{5.4}$$

where c_t is the context vector calculated by a weighted sum of the encoder hidden states.

$$c_i = \sum_{j=1} \alpha_{ij} h_j \tag{5.5}$$

where α_{ij} is the amount of attention the ith output should pay to the jth input and hj is the encoder state for the jth input. α_{ij} is computed by taking a softmax over the attention scores **e** of the inputs with respect to the ith output.

$$\alpha_{ij} = \sum_{j=1} \alpha_{ij} h_j \tag{5.6}$$

$$\alpha_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k=1} exp(e_{ik})} \tag{5.7}$$

where

$$e_{ij} = f(s_{i-1}, h_j) \tag{5.8}$$

Algorithm 2: Pseudo-code for the REINFORCE algorithm applied to encoder-decoder

```
Input: A sequence of input tokens(X)
Output: ground truth MeSH sequences(y)
while not converged do
    Sample N batches from X and y
    Encode the input using the Encoder RNN
    Initialize the current token with the start token T= '<START>'
    Initialize the output sequence with empty sequence: Out = []
    while T != '<END>' do
        Get the probability distribution of the output tokens
        Sample the output token B_out from the distribution
        Add B_out to the output sequence: Out+ = B_out
        Set the current token T
    end
    Calculate the Q = F1-score
    Estimate the gradients ∇j = Σ_T Q∇ log p(T)
    Update the model using SGD
end
```

5.3.3 Reinforcement learning for seq2seq training

To alleviate the problems with the Encoder-Decoder model (exposure bias, train/test measure inconsistency) as well as to prevent the high dependence of the Encoder-decoder model on the label order, we can model the task as a reinforcement learning (RL) problem.

The first thing to note is that the decoder outputs the next token probability distribution at every step, which is very similar to policy gradient models [72]. From this perspective, the decoder can be seen as an agent trying to decide which token to produce at every step. This stochastic framing helps to take into account multiple target sequences and learn how to produce multiple variants of the same sequence with different ordering. Another advantage of this stochastic framing is that we can directly minimize the F1-score we care about in inference, instead of minimizing the cross-entropy. Even when the reward (F1-score) is not differentiable, we can use gradient methods like REINFORCE [84] to push up the probabilities of successful

episodes and decrease the worse ones. The working of the REINFORCE algorithm applied to Encoder-Decoder models is shown in Algorithm 2.

Thus, the training objective is to minimize the negative expected reward:

$$L(\theta) = -E_{y \sim p\theta}[r(\mathbf{y})] \tag{5.9}$$

We used the F1 score as the reward function by comparing the generated labels with the ground truth.

5.4 Experimental Results

5.4.1 Dataset

We use the PubMed dataset which contains over 10 million annotated PubMed articles. The dataset includes 28,789 MeSH terms in total. Each article in the dataset is annotated with 12 MeSH terms on average. We performed our training on 3 million articles. Another 100,000 articles are used for testing. Our model is implemented using tensorflow 2.4 with Adam optimizer and batch size of 128. We employed early stopping and learning rate decay strategies. To train the model on 4 GPUS (V100s) takes 2 days.

5.4.2 Evaluation and Results

The performance of our model is evaluated by the harmonic mean of micro-precision (MiP) and micro-recall (MiR), Micro-F. The Micro-F is calculated as:

$$Micro - F = \frac{2 \cdot MiP \cdot MiR}{MiP + MiR} \tag{5.10}$$

where

$$MiP = \frac{\sum_{i=1}^{N_a}\sum_{j=1}^{N} y_{ij} \cdot \hat{y}_{ij}}{\sum_{i=1}^{N_a}\sum_{j=1}^{N} \hat{y}_{ij}} \tag{5.11}$$

$$MiR = \frac{\sum_{i=1}^{N_a}\sum_{j=1}^{N} y_{ij} \cdot \hat{y}_{ij}}{\sum_{i=1}^{N_a}\sum_{j=1}^{N} y_{ij}} \tag{5.12}$$

where N_a is the the total number of test articles, N is the number of all MeSH terms, i is index for articles and j is index for MeSH terms. y_{ij} denotes whether MeSH term j is in article i in the ground truth, and $\hat{y}_{ij}$ denotes whether MeSH term j is in article i in the prediction.

The performance comparisons of our model and baselines is shown in Table 5.1. As can be seen from the results, modeling MeSH indexing task as a seq2seq problem can improve performance.

Model	**MiP**	**MiR**	**MiF**
MTI First Line Index	0.6	0.65	0.63
DeepMeSH	0.65	0.6	0.63
MeSHNow	0.61	0.61	0.61
AttentionMeSH	0.68	0.65	0.66
Our model	0.66	0.68	0.67

Table 5.1: Average MiF scores for the baselines and our model.

We also experimented with how the gains from our model compare against the best performing baseline model in different categories. By binning labels into the number of supporting training documents each label has, table 5.2 shows the gains in each bin. We can see that as the number of supporting documents a label has decreases, the gain from our model increases, hence the benefit of our architecture.

As a case study, we probe our model to determine what kind of MeSH sequences

Model	≥ 5000	≥ 1000	≥ 500	≥ 100	≥ 25	≤ 25
Our model	0.69	0.61	0.58	0.57	0.46	0.18
AttentionMeSH	0.69	0.59	0.58	0.58	0.42	0.15

Table 5.2: Average MiF scores for our model and AttentionMesh using bins.

are challenging. We show four example MeSH sequences in Figure 5.4. The two example sequences on the left are where our model was able to correctly generate the labels "Rhodococcus equi" and "Perfusion imaging" even though these labels occur fewer than 25 times in the training set. This is due to the fact that the preceding labels in the sequence are closely related to the rarely occurring labels and our model was able to learn the relationship. In contrast, the example sequences on the right were difficult for our model and the two rarely occurring labels "Academic Medical Centers" and "Poland" are not generated. We hypothesize this is because the preceding labels in each sequence do not provide enough information to predict the rarely occurring labels.

In Table 5.3, we calculate the bootstrap sample statistic of the Micro F1 score using Monte Carlo simulation. We performed the bootstrap procedure 1000 times by drawing samples of 10,000 abstracts from the dataset with replacement. The Micro F1 score is the statistic on which we estimated using bootstrap samples. We can see that there is a 95% likelihood that the range 0.666 to 0.675 contains the true statistic mean (Micro F1 score) suggesting the performance improvements we gain are significant over the baseline model AttentionMesh.

Model	**2.5th percentile**	**50th percentile(median)**	**97.5th percentile**
AttentionMesh	0.657	0.661	0.665
Our Model	0.666	0.671	0.675

Table 5.3: Bootstrap sample statistic using 1000 Monte Carlo simulations.

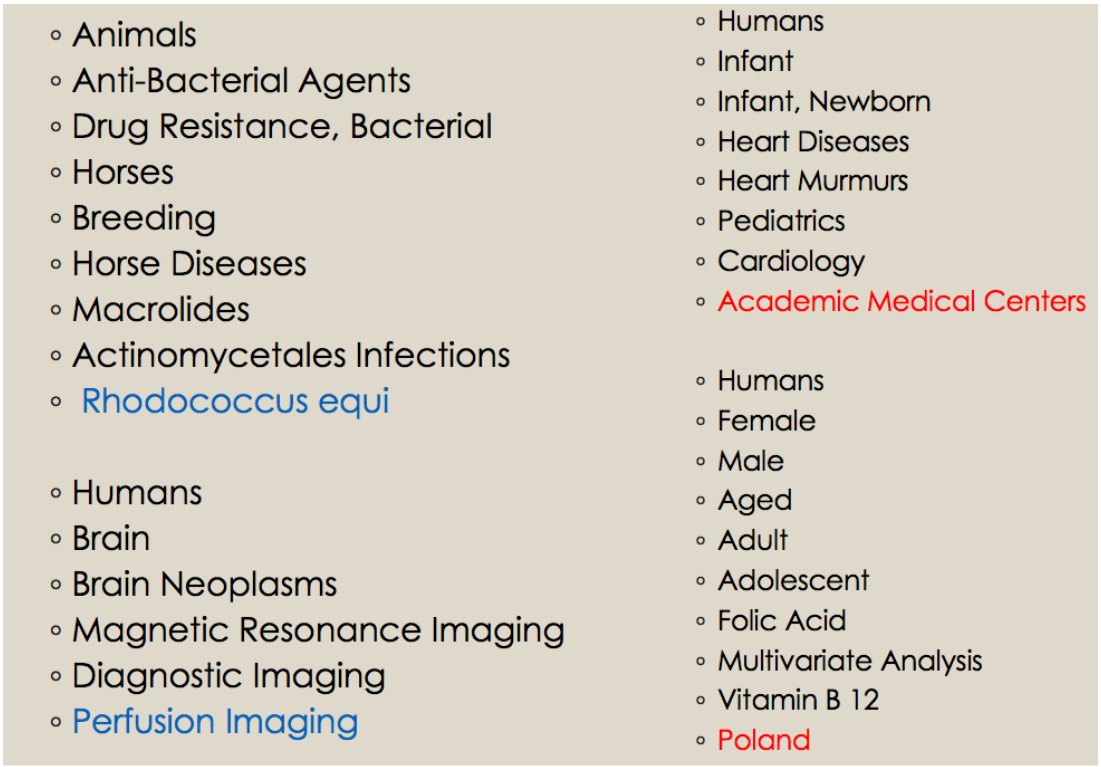


Figure 5.4: Examples of MeSH sequences. The left side are two cases where our model does well while the right side illustrates the challenging examples.

5.5 Conclusion

In this chapter, we proposed a reinforcement based encoder-decoder model for the task of MeSH indexing. We trained an attention-based encoder-decoder model by reordering the MeSH terms as sequences by their frequency of occurrence. Since the input sequence and output sequence vary widely in the number of tokens, the employment of a large attention block is of paramount importance. To further improve limitations within the encoder-decoder architecture, we fine-tune the model with reinforcement learning using the REINFORCE algorithm which allows us to optimize for F1-score instead of the log-loss. The empirical results on the PubMed dataset showcase that using reinforcement based encoder-decoder model can provide an performance improvement over baseline methods which use variations of multi-label classification.

Chapter 6

Conclusion and Future work

In this dissertation, we focused on developing machine learning models to solve the related problems of keyphrase extraction and MeSH Indexing. Although many machine learning based models have been proposed to solve the tasks, there still is room for further improvement.

The first three works focused on author provided keyphrase extraction from biomedical documents while the last work is on MeSH indexing. In the first work, we introduced NamedKeys – a method composed of four modules: NER, phrase embedding, phrase quality score and similarity-based clustering for keyphrase extraction from biomedical documents. To evaluate the proposed method, we created a new publicly available benchmark dataset from PubMed Central Open Access articles. We showed that this method has performance gains over other unsupervised methods.

In the second work, we proposed a keyphrase extraction method that focuses on identifying words which are central to the document semantics. The problem of keyphrase extraction is posed as a sequence labeling task where each token is tagged as either a keyphrase or not. In addition to our novel centrality constraint layer, we used BiLSTM layers to capture the long term dependencies among the input sequences. Finally, we have a CRF layer which is well suited to capture the dependencies from

the output labels. Empirical results on two datasets shows that our method gains significant improvement in the PubMed dataset while performing slightly better on the INSPEC dataset.

A semi-supervised method was proposed in our third work where a new uncertainty-based self-training keyphrase extraction method is employed that utilizes unlabeled data to augment small labeled training data. We proposed the use of Monte-Carlo dropout to approximate the model uncertainty for each pseudo-labeled document. The uncertainty is then used to sample specific documents to retrain the model using the combined data. This iterative Teacher-Student model training is performed until convergence is achieved. The empirical results on the PubMed dataset showcase that self-training can provide an performance improvement, especially when there is a significant unlabeled corpus.

Finally, in the last work, we proposed a reinforcement based encoder-decoder model for the task of MeSH indexing. We trained attention based encoder-decoder model by reordering MeSH terms as sequences by their frequency of occurrence. To further improve limitations within the encoder-decoder architecture, we fine-tuned the model with reinforcement learning using the REINFORCE algorithm to optimize the F1-score instead of the log-loss. The empirical results on the PubMed dataset showcase that using reinforcement based encoder-decoder model can provide an performance improvement over baseline methods which use variations of multi-label classification.

We believe there are ways to further improve the proposed methods. For the task of keyphrase extraction, an obvious potential direction is to include keyphrase abstraction. In this work, we focused on keyphrases that exist in the abstract and title of biomedical documents. However, authors usually provide keys that are not in the abstract and title. Even though the evaluation of such methods is harder, abstractive extraction has more potential especially given the success of text generation models

such as BERT.

For MeSH indexing, we can expand the work by using Transformer based Encoder-Decoder model which may handle the long distance relation between tokens better. Another potential direction is to merge the two tasks of keyphrase extraction and MeSH indexing. Under this setting, an end-to-end model can be designed that generates the MeSH terms for an input abstract and use those MeSH terms as an additional input to extract keyphrases.

Finally, another direction that we may consider relates to the fact that the gold true labels we have used throughout the experiments for keyphrase extraction are keys submitted by authors. We believe better keyphrases may be extracted if we train on keyphrases manually annotated by an expert instead of using various authors' keys.

Bibliography

[1] Rabah Alzaidy, Cornelia Caragea, and C Lee Giles. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The world wide web conference*, pages 2551–2557, 2019.

[2] Alan R Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association, 2001.

[3] Alan R Aronson, James G Mork, Francois-Michel Lang, Willie J Rogers, and Aurelie Neveol. Nlm medical text indexer: A tool for automatic and assisted indexing. *Bethesda: US National Library of Medicine*, 2008.

[4] Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853*, 2017.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *Proc. of EMNLP*, pages 3615–3620, 2019.

[7] Santosh Kumar Bharti and Korra Sathya Babu. Automatic keyword extraction for text summarization: A survey. *arXiv preprint arXiv:1704.03242*, 2017.

[8] Sanmitra Bhattacharya, Viet Ha-Thuc, and Padmini Srinivasan. Mesh: a window into full text for document summarization. *Bioinformatics*, 27(13):i120–i128, 2011.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[10] Willie Boag, Elena Sergeeva, Saurabh Kulshreshtha, Peter Szolovits, Anna Rumshisky, and Tristan Naumann. Cliner 2.0: Accessible and accurate clinical concept extraction. *arXiv preprint arXiv:1803.02245*, 2018.

[11] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[12] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

[13] Florian Boudin. pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 69–73, Osaka, Japan, December 2016. URL `http://aclweb.org/anthology/C16-2015`.

[14] Florian Boudin. Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*, 2018.

[15] Adrien Bougouin, Florian Boudin, and Béatrice Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 543–551, 2013.

[16] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer, 2018.

[17] Jason Chuang, Christopher D Manning, and Jeffrey Heer. "without the clutter of unimportant words": Descriptive keyphrases for text visualization. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(3):19, 2012.

[18] Young Mee Chung and Jae Yun Lee. A corpus-based approach to comparative evaluation of statistical term association measures. *Journal of the American Society for Information Science and Technology*, 52(4):283–296, January 2001.

[19] Frans Coenen, Paul Leng, Robert Sanderson, and Yanbo J Wang. Statistical identification of key phrases for text classification. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 838–853. Springer, 2007.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[21] Samhaa R El-Beltagy and Ahmed Rafea. Kp-miner: Participation in semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 190–193, 2010.

[22] Corina Florescu and Cornelia Caragea. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, 2017.

[23] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[24] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of ICML*, pages 1050–1059, 2016.

[25] Zelalem Gero and Joyce C. Ho. Pmcvec: Distributed phrase representation for biomedical text processing. *Journal of biomedical Informatics, in press*, 2019.

[26] Zelalem Gero and Joyce C Ho. Namedkeys: Unsupervised keyphrase extraction for biomedical documents. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 328–337, 2019.

[27] Zelalem Gero and Joyce C Ho. Uncertainty-based self-training for keyphrase extraction. In *Proceedings of the 2021 IEEE EMBS International Conference on Biomedical & Health Informatics*, 2021.

[28] Zelalem Gero and Joyce C Ho. Word centrality constrained representation for keyphrase extraction. In *BioNLP: Workshop on Biomedical Natural Language Processing*, 2021.

[29] Glove vec. Glove: Global vectors for word representation. `https://nlp.stanford.edu/projects/glove/`.

[30] Google. word2vec: Tool for computing continuous distributed representations of words. `https://code.google.com/archive/p/word2vec/`.

[31] Jun Gu, Wei Feng, Jia Zeng, Hiroshi Mamitsuka, and Shanfeng Zhu. Efficient semisupervised medline document clustering with mesh-semantic and global-content constraints. *IEEE transactions on cybernetics*, 43(4):1265–1276, 2012.

[32] Kazi Saidul Hasan and Vincent Ng. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1262–1273, 2014.

[33] Minlie Huang, Aurélie Névéol, and Zhiyong Lu. Recommending mesh terms for annotating biomedical articles. *Journal of the American Medical Informatics Association*, 18(5):660–667, 2011.

[34] Xiaodi Huang, Xiaodong Zheng, Wei Yuan, Fei Wang, and Shanfeng Zhu. Enhanced clustering of biomedical documents using ensemble non-negative matrix factorization. *Information Sciences*, 181(11):2293–2302, 2011.

[35] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223, 2003.

[36] Aminul Islam, Evangelos E Milios, and Vlado Keselj. Comparing word relatedness measures based on Google *n*-grams. In *Proceedings of COLING 2012: Posters*, pages 495–506, 2012.

[37] Xin Jiang, Yunhua Hu, and Hang Li. A ranking approach to keyphrase extraction. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 756–757. ACM, 2009.

[38] Antonio Jimeno-Yepes, James G Mork, Dina Demner-Fushman, and Alan R Aronson. A one-size-fits-all indexing method does not exist: automatic selection based on meta-learning. *Journal of Computing Science and Engineering*, 6(2): 151–160, 2012.

[39] Antonio Jimeno Yepes, James G Mork, BartBomiej Wilkowski, Dina Demner Fushman, and Alan R Aronson. Medline mesh indexing: lessons learned

from machine learning and future directions. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 737–742, 2012.

[40] Antonio J Jimeno-Yepes, Bridget T McInnes, and Alan R Aronson. Exploiting mesh indexing in medline to generate a data set for word sense disambiguation. *BMC bioinformatics*, 12(1):1–14, 2011.

[41] Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. Attentionmesh: simple, effective and interpretable automatic mesh indexer. In *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*, pages 47–56, 2018.

[42] Su Nam Kim and Min-Yen Kan. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the workshop on multiword expressions: Identification, interpretation, disambiguation and applications*, pages 9–16. Association for Computational Linguistics, 2009.

[43] G Hemantha Kumar, Seyedmahmoud Talebi, and K Manoj. Users' topic detection from tweets based on keyword extraction. *International Journal of Computer Applications*, 975:8887, 2017.

[44] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[45] Tuan Manh Lai, Trung Bui, Doo Soon Kim, and Quan Hung Tran. A joint learning approach based on self-distillation for keyphrase extraction from scientific documents. In *Proc. of the 28th International Conference on Computational Linguistics*, pages 649–656, 2020.

[46] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language

representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[47] Quanzhi Li and Yi-Fang Brook Wu. Identifying important concepts from medical documents. *Journal of biomedical informatics*, 39(6):668–679, 2006.

[48] Xinzhe Li, Qianru Sun, Yaoyao Liu, Shibao Zheng, Qin Zhou, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. In *Proc. of NeurIPS*, pages 10276–10286, 2019.

[49] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 366–376. Association for Computational Linguistics, 2010.

[50] Zhiyong Lu. Pubmed and beyond: a survey of web tools for searching biomedical literature. *Database*, 2011, 2011.

[51] Zhiyong Lu, Won Kim, and W John Wilbur. Evaluation of query expansion using mesh in pubmed. *Information retrieval*, 12(1):69–80, 2009.

[52] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[53] Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. 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 634–639, 2018.

[54] Yuqing Mao and Zhiyong Lu. Mesh now: automatic mesh indexing at pubmed scale via learning to rank. *Journal of biomedical semantics*, 8(1):1–9, 2017.

[55] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[56] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[57] James G Mork, Antonio Jimeno-Yepes, Alan R Aronson, et al. The nlm medical text indexer system for indexing biomedical literature. *BioASQ@ CLEF*, 1, 2013.

[58] Subhabrata Mukherjee and Ahmed Hassan Awadallah. Uncertainty-aware self-training for text classification with few labels. In *Proc. of NeurIPS*, pages 21199–21212, 2020.

[59] Naw Naw and Ei Ei Hlaing. Relevant words extraction method for recommendation system. *Bulletin of Electrical Engineering and Informatics*, 2(3):169–176, 2013.

[60] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. Scispacy: Fast and robust models for biomedical natural language processing. *arXiv preprint arXiv:1902.07669*, 2019.

[61] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[62] Aditya Parameswaran, Hector Garcia-Molina, and Anand Rajaraman. Towards

the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 0(1-2):566–577, 2010.

[63] Shengwen Peng, Ronghui You, Hongning Wang, Chengxiang Zhai, Hiroshi Mamitsuka, and Shanfeng Zhu. Deepmesh: deep semantic representation for improving large-scale mesh indexing. *Bioinformatics*, 32(12):i70–i79, 2016.

[64] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[65] Vahed Qazvinian, Dragomir R Radev, and Arzucan Ozgur. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd international conference on computational linguistics (COLING 2010)*, pages 895–903, 2010.

[66] Wullianallur Raghupathi and Viju Raghupathi. Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2(1):3, 2014.

[67] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

[68] Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. Keyphrase extraction as sequence labeling using contextualized embeddings. In *European Conference on Information Retrieval*, pages 328–335. Springer, 2020.

[69] Tokala Yaswanth Sri Sai Santosh, Debarshi Kumar Sanyal, Plaban Kumar Bhowmick, and Partha Pratim Das. Dake: Document-level attention for keyphrase extraction. In *European Conference on Information Retrieval*, pages 392–401. Springer, 2020.

[70] Kamal Sarkar. A hybrid approach to extract keyphrases from medical documents. *arXiv preprint arXiv:1303.1441*, 2013.

[71] Kamal Sarkar. A keyphrase-based approach to text summarization for english and bengali documents. *International Journal of Technology Diffusion (IJTD)*, 5(2):28–38, 2014.

[72] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[73] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[74] Jafar Tanha, Maarten van Someren, and Hamideh Afsarmanesh. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8(1):355–370, 2017.

[75] Stamatina Thomaidou and Michalis Vazirgiannis. Multiword keyword recommendation system for online advertising. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 423–427. IEEE, 2011.

[76] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment*, 2003.

[77] Dolf Trieschnigg, Piotr Pezik, Vivian Lee, Franciska De Jong, Wessel Kraaij, and Dietrich Rebholz-Schuhmann. Mesh up: effective mesh text classification for improved document retrieval. *Bioinformatics*, 25(11):1412–1418, 2009.

[78] Peter D Turney. Learning algorithms for keyphrase extraction. *Information retrieval*, 2(4):303–336, 2000.

[79] Peter D Turney. Learning to extract keyphrases from text. *arXiv preprint cs/0212013*, 2002.

[80] Xiaojun Wan and Jianguo Xiao. Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 969–976. Association for Computational Linguistics, 2008.

[81] Xiaojun Wan and Jianguo Xiao. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860, 2008.

[82] Rui Wang, Wei Liu, and Chris McDonald. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, volume 39, 2014.

[83] Christian Wartena and Rogier Brussee. Topic detection by clustering keywords. In *2008 19th International Workshop on Database and Expert Systems Applications*, pages 54–58. IEEE, 2008.

[84] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[85] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI Global, 2005.

[86] Zhao Yan, Duyu Tang, Nan Duan, Shujie Liu, Wendi Wang, Daxin Jiang, Ming Zhou, and Zhoujun Li. Assertion-based qa with question-aware open information

extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[87] Wen-tau Yih, Joshua Goodman, and Vitor R Carvalho. Finding advertising keywords on web pages. In *Proceedings of the 15th international conference on World Wide Web*, pages 213–222. ACM, 2006.

[88] Hanyu Zhang, Che-Lun Hung, Meiyuan Liu, Xiaoye Hu, and Yi-Yang Lin. Nc-net: Deep learning network models for predicting function of non-coding dna. *Frontiers in genetics*, 10:432, 2019.

[89] Qi Zhang, Yang Wang, Yeyun Gong, and Xuan-Jing Huang. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 836–845, 2016.

[90] Yanchun Zhang, S Peng, R You, Z Xie, B Wang, and Shanfeng Zhu. The fudan participation in the 2015 bioasq challenge: Large-scale biomedical semantic indexing and question answering. In *CEUR Workshop Proceedings*, volume 1391. CEUR Workshop Proceedings, 2015.

[91] Xun Zhu, Chen Lyu, Donghong Ji, Han Liao, and Fei Li. Deep neural model with self-training for scientific keyphrase extraction. *Plos one*, 15(5):e0232547, 2020.