UNIVERSITY OF AMSTERDAM

MSc ARTIFICIAL INTELLIGENCE
MASTER THESIS

Supervised Neural Disease Normalization

by

DHRUBA PUJARY

11576200

August 22, 2019

36 EC
January 2019 - August 2019

Supervisor:
Dr. W. FERREIRA AZIZ (UvA)
Dr. C. THORNE (Elsevier)

Assessor:
Dr. M. A. RIOS GAONA (UvA)

UvA, INSTITUTE FOR LOGIC,
LANGUAGE AND COMPUTATION

CONTENT TRANSFORMATION,
LIFE SCIENCES

Abstract

A key task in biomedical text mining and in particular, content enrichment, consists in identifying objects of interest or entities – named entity recognition – and normalizing (disambiguating) named entities against knowledge repositories. Such task is known in the wider natural language processing (NLP) area as entity linking, and has given rise to a number of powerful techniques that can link with high performance open domain entities to encyclopedic knowledge graphs. In the bio-medical domain however, knowledge repositories tend to be flat and of a more modest scale. Corpora also exhibit different linguistic properties (style, structure, vocabulary). We investigate and adapt recent advancements in neural entity linking, leveraging embedding-based encodings of both the source and target entities, to the task of disease normalization against the Medical Subject Headings (MeSH) taxonomy. We leverage the graph structure and the large content of MeSH to generate graph embeddings based on models such as node2vec (Grover and Leskovec, 2016) and compositional functions such as Graph Convolutional Networks (Kipf and Welling, 2016). We investigate both entity recognition (a sequence labelling task) and entity linking (a classification task), and evaluate our approaches using standard metrics such as precision, recall, F1-score and accuracy and analyze how information is encoded in each of our graph based embeddings. Our findings suggest that using ELMo (Peters et al., 2018) embeddings for word representation and node lexicalization results in improved performance in EL task as compared to our baseline. However, the results do not improve over other known methods (Leaman et al., 2013, DNorm) in disease normalization. A natural follow-up question is how to improve the node representations by incorporating more lexical information and/or the hierarchical structure of the diseases for the classification task.

Acknowledgements

I would like to thank Wilker Aziz for his generous supervision and guidance. His excellent practical advice on the different method implementation, as well as detailed theoretical grounding of new ideas, kept me motivated every day.

I would like to thank Camilo Thorne for introducing me to this interesting topic and his supervision throughout the thesis has helped a lot. Our interesting discussions have been very useful and working with him turned out to be a great learning experience. I am very grateful to Elsevier, where I worked on parts of this thesis as an intern under Camilo's supervision at the Content Transformation, Life Sciences division, as part of the ongoing collaboration between UvA and Elsevier.

Thanks to my committee Miguel A. Rios Gaona, Wilker Aziz and Camilo Thorne for agreeing to asses my work.

Finally, I express my very profound gratitude to my family and to my friends for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Contents

Chapter 1

Introduction

With a plethora of information available in the form of web pages, documents, articles, etc. at our disposal, making efficient use of these resources is crucial. Natural Language Processing (NLP) is a sub-field of computer science, information science, and artificial intelligence which is concerned with utilizing computers to process a huge amount of human or natural language data to turn it into information usable by a computer. An NLP based system can extract important information in less amount of time and minimal human intervention. Fast progress in NLP can improve many other domains of research, like the bio-medical domain which also has a massive amount of data. For instance, a laboratory report in the form of a textual document of a patient may contain mentions of many medical terms and diagnosis results. We can identify the most probable diseases by utilizing all previous reports of different patients and recognizing the patterns of co-occurrence of similar symptoms or diagnosis results. Machine learning is most suited for recognizing a pattern in data and building a probabilistic model of the data generation process. The field of NLP research using machine learning algorithms has advanced a lot in recent years. Some of the researched tasks are machine translation, sentiment analysis, question answering, etc. Commercial applications using NLP also have a lot of impact , for example, a machine translation system trained on a set of parallel text of two different languages, say German and English that can translate sentence in one language to another or vice versa (Edunov et al., 2018), will allow communication between any two-person (or parties). The Bio-medical domain has also incorporated the use of NLP in many of its applications , for instance, disease recognition and normalization, drug discovery, medical question answering, medical auto-completion , etc.

In bio-medical research, many papers, articles, journals, etc. are published every year and keeping track of every new discovery, improvement or change is required. This quantity is so large, that it cannot be done manually. One of the many problems addressed using NLP is identifying disease names or drug discovery. Named Entity Recognition (NER) is the first step of many of the NLP-based systems for subsequent information search tasks such as question answering, etc. The term *"Named Entity"* was coined in the Sixth Message Understanding Conference (Grishman and Sundheim, 1996, MUC-6) and is widely used in the NLP community . NER is the task to identify any information unit of interest in a given text. The information unit or *"Entity"* (mention) could be the name of a person, location or organization. Intuitively, any expression referring to an object of interest in a domain of a given kind. In the bio-medical domain, NER is used to find mentions such genes, proteins, diseases, drugs, and organisms in natural language text. Also, bio-medical NER is considered to be more difficult than other domain, such as in CoNLL-2002 shared task (Tjong Kim Sang and De Meulder, 2003). First, there are millions of entity names in use and new ones are added constantly, implying that neither dictionaries nor training data will be sufficiently comprehensive. Second, the bio-medical field is moving quickly to build a consensus on the name to be used for a given entity or even the exact concept defined by the entity itself. Sometimes authors use very similar or even identical names and acronyms for different concepts and prefer to introduce their own abbreviation and use throughout the paper regardless of naming conventions. Finally, entity names in bio-medical text are longer on average than names from other domain.

NER step allows us to identify the entities in a document but it may have multiple meanings or refer to different information depending on the context of usage. For example, the name “Diabetes" can be used to denote different kind of diabetes (e.g., I or II). The naive way of disambiguation would be to maintain a record of all the properties of Diabetes Type-I and Type-II and disambiguate against every such record. These records would be very large and keep growing as new discoveries happen

every day. To manage such large quantity of information, people have invested on building data or knowledge repositories known as Knowledge Bases (KBs) to segregate and organize the information in well structured and meaningful way (Auer et al., 2008; Bollacker et al., 2008; Suchanek et al., 2007). The use of hierarchical segregation of information or concepts is known as a *Taxonomy*. Information in a taxonomy is arranged in a manner where generic information are placed at the top in the hierarchy and with each level, the information are classified based on some distinguishing characteristics. For example, animals would be at the top in the hierarchy which can be classified into herbivore, carnivore and omnivore based on what animals consumes.

One of the critical steps to leverage the benefits of having a huge amount of raw data and large-scale machine-readable KBs is to link the named entity with their corresponding concepts in the KB, which is called Entity Linking (EL) or Normalization (a form of disambiguation). The EL task is challenging due to name variations and entity ambiguity (Shen et al., 2015). A named entity may have multiple surface forms, such as its full name, partial names, aliases, abbreviations, and alternate spellings. However, an NLP-based system should be able to learn a mapping efficiently rather than comparing a named entity to every other entity in KB.

In the bio-medical domain, disease normalization is the task of identifying the disease names in a document and mapping them to a unique concept in the KB. For identification, we first perform disease NER and subsequently EL for the mapping. An automated system depending on these upstream tasks can help in the progress of bio-medical research. For example, articles and papers which are used by the researchers as well as medical professionals can be indexed more accurately for identifying cures, properties, drugs, related to query, etc. The efficiency of these automated system relies on the algorithms used for the NER and EL. Until recently, most of the algorithms were rule-based or classical hand-engineered features based machine learning. Advances in Neural Network (NN) architectures, especially Deep Learning (DL), has led to many avenues of research in NLP. However, domain adaptation of the techniques and architectures developed in DL and related methods are yet to be explored as all of these are making progress very rapidly. Moreover, disease normalization has never been approached in this domain , to the best of our knowledge, by taking into account the taxonomical structure of the target KB. Apart from the semantic information present in the description, the taxonomical structure implicitly encapsulates discriminative information which could be used for linking. For example, in a balanced binary tree with three nodes, child nodes will have a certain quantity of information similar to or shared with their parent, but also information that discriminates between the siblings.

1.1 Research and Contribution

In this thesis, we seek to address the following **research questions**:

1. **How can we adapt state-of-the-art open-domain neural architectures for NER and entity linking to the disease domain?** Most of the research applying deep learning are open-domain and its adaptation is very limited in other domain. A well-known disease normalization system known as DNorm (Leaman et al., 2013)) uses Banner NER[1] which is almost a decade old. We explore the latest state-of-the-art models and adapt these open domain neural architectures to disease domain. We consider Lample et al. (2016) and Ma and Hovy (2016) as our baseline models and explore various changes.

2. **What kind of embeddings and combinations thereof (word or character, pre-computed, domain- or corpora-specific) are the most promising for this task?** Contextualized word embedding such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2018) has recently outperformed in few text classification NLP task. We explore ELMo in our models and compare with word2vec by Mikolov et al. (2013). Similarly, for EL, we exploit the taxonomy using graph embeddings and develop architectures to aid our disambiguation task.

3. **Can neural networks improve also on classical methods for disease disambiguation?** We consider different metrics for evaluating our models and doing a fair comparison among each. We use standard evaluation metric for the NER system, namely precision, recall, and F1-score. Our EL approach is novel and different so we consider accuracy as our metric and use different relaxation strategy based on rank to evaluate the quality of our models.

[1] http://banner.sourceforge.net/

4. **Can an end-to-end or multi-task neural architectures, combining NER and EL in a single network improve on a pipeline architecture?** Most known systems for normalization are usually pipeline architectures, i.e, we sequentially perform NER followed by EL which propagates the error of former system directly to later. Moreover, since the EL task is dependent on the NER task, we could use the signal common to both the task to mutually benefit each other. Multi-task learning is the most suitable candidate in such a scenario. The use of multi-task learning (MTL) in machine learning application has been numerous, from natural language processing to computer vision where we optimize more than one loss function. MTL is beneficial for training multiple tasks that are similar because it ignores the data-dependent noise by averaging the noise patterns of different task and helps in generalization. MTL can also help the model to focus its attention on features that are relevant for a task as other tasks provide additional evidence for the relevance or irrelevance of features in situations when the task is noisy or data is limited and high-dimensional. In some cases, it might even help to learn features relevant for a task but difficult to extract using the same task rather its easy to extract by training on a different task.

Our contribution in this thesis are the following:

1. **Taxonomy Embedding using Graph Neural Networks**: Previous implementation of disease normalization never considered the use of the taxonomy of diseases in their models to best of our knowledge. We propose a novel approach that uses graph based neural networks incorporating the taxonomy information as well as the description of the diseases.
2. **Multi-task Learning with attention for disease recognition and classification**: We study the behaviour of attention mechanism in our MTL model with different activation function which provides insights on how the NER and EL tasks influence each other.
3. **State of the art disease recognition (NER)**: We adapt state of the art neural NER models to the disease domain, and achieve results on a par with current (as of 2019)[2] results for this domain.

1.2 Outline of the thesis

The rest of the thesis is structured as follows:

1. Chapter 2 deals with the related work and theoretical background required in this thesis.
2. Chapter 3 describes the dataset and the knowledge base used for normalization.
3. Chapter 4 describes in detail the task of disease recognition, the model architectures and evaluation procedure along with the results.
4. Chapter 5 describes in detail the disease normalization task, the encoding of taxonomical information and various architectures along with evaluation and results.
5. Chapter 6 describes in detail multi-task learning for disease normalization where we explore different activation function and results.
6. Chapter 7 briefly describes the conclusion of the thesis and outlines of future research works in this field.

[2] https://paperswithcode.com/sota/named-entity-recognition-ner-on-ncbi-disease

Chapter 2

Background and related work

2.1 Related work

In this section, we provide a brief review of research done for NER 2.1.1, EL 2.1.2 and learning jointly both tasks in 2.1.3. In each subsection, we first present models and improvements used in general NLP research followed by bio-medical domain-specific methods.

2.1.1 Named Entity Recognition

NER has been approached in the past as supervised, semi-supervised and unsupervised machine learning problems. Supervised learning algorithms include Hidden Markov Models (Bikel et al., 1999, HMM), Decision Trees, Maximum Entropy Models (Borthwick, 1999; Curran and Clark, 2003, ME), Support Vector Machines (McNamee and Mayfield, 2002, SVM) and Conditional Random Fields (McCallum and Li, 2003, CRF). Due to the lack of annotated data and data sparsity problems, semi-supervised algorithms such as AdaBoost (Carreras et al., 2002). Unsupervised approaches includes NER systems applying generic pattern matching (Etzioni et al., 2005), clustering (Nadeau and Sekine, 2007) and bootstrapping(Munro and Manning, 2012).

Collobert and Weston (Collobert and Weston, 2008) proposed one of the first neural network architectures for NER with manually constructed feature vectors from dictionaries, lexicons and orthographic features (e.g. capital letters) and later replaced them with word embeddings (Collobert et al., 2011). Their proposed model would take sentences as inputs, learn or use pre-trained word embeddings and extract higher-level features from the word embedding vectors by using convolutional layers followed by fully connected layers. The output dimension of the (last) affine layer is the same as the total number of tags and is fed to CRF (Lafferty et al., 2001) classifier for final prediction. dos Santos and Guimarães (2015) extended Colbert et al.'s work with character-level representation using CNN to extract morphological information like prefix or suffix of a word. Huang et al. (2015) proposed a word-level bidirectional Long Short Term Memory network (Hochreiter and Schmidhuber, 1997) (Bi-LSTM) with a CRF classifier (BiLSTM-CRF) which is less dependent on word embeddings as compared to the previous models. Extending this basic architecture further Lample et al. (2016) included a character-level embedding trained using a variant of the BiLSTM model. The use of character embeddings is motivated by the fact that important features in languages are position-dependent (e.g., prefixes and suffixes encode different information than stems). Kuru et al. (2016) model takes only character-level input i.e sentences are represented as a sequence of characters instead of words, which consist of stacked Bi-LSTM and outputs tags probabilities for each character. Using a Viterbi decoder, these probabilities are then converted to consistent word-level named entity tags. Ma and Hovy (2016) used a similar architecture as Lample et al. (2016) except they use CNN for character-level embedding instead of Bi-LSTMs. The CNN approach takes only trigrams into account and are position-independent, whereas the Bi-LSTM is position aware. Nevertheless, Reimers and Gurevych (2017) found that using either character representation is statistically insignificant for NER.

Early bio-medical named entity recognition approaches used rule-based dictionary matching methods which require features to some extent (Lin et al., 2004; Jimeno et al., 2008; Lowe and Sayle, 2015). As CRFs became widely used in sequence labeling tasks, BANNER (Leaman, 2008) and Sun et al. (2007) used CRFs with feature engineering for NER. Sahu and Anand (2016) proposed various end-to-end recurrent neural networks models removing the dependency on hand-crafted features. Habibi et al. (2017)

used Lample et al. (2016) model architecture of BiLSTM-CRF and word embeddings. Instead of treating disease NER as a sequence tagging problem, Zhao et al. (2017) approached it as a word-level classification problem assuming that the contextual information is enough to predict the target word's label. They used multiple label CNNs to capture the context information within a fixed-size window of each target word.

Our baseline implementation of Lample et al. (2016) is different from the original implementation in terms of the LSTM architecture. They use the Gers and Schmidhuber (2000) implementation whereas we use the original implementation by Hochreiter and Schmidhuber (1997). Also, our baseline implementation of Ma and Hovy (2016) differs from the original in terms of the CNN layer. We use two CNN layers whereas the original author used only one. On top of these modifications, we test various other recent techniques for improvement in performance.

2.1.2 Entity Linking

Early work in EL (or Entity Disambiguation) was focused on finding sophisticated ways to measure the relatedness between the context of a mention and the definition of the candidate entities (C. Bunescu and Pasca, 2006; Kulkarni et al., 2009; Zheng et al., 2010; Hoffart et al., 2011; Zhang et al., 2011). Another approach focused on collective disambiguation where mentions within the same context are resolved simultaneously based on the coherence of the decisions (Kulkarni et al., 2009; Hoffart et al., 2011; Ratinov et al., 2011; Han et al., 2011). He et al. (2013) proposed a deep neural network model to compare context and entity at some higher-level abstraction with general concepts getting shared across entities at lower levels. Specifically, they used Stacked Denoising Auto-encoders to discover general encoding of documents containing the mentions, as well as of candidate entities in KB entries in an unsupervised fashion, topped by a supervised fine-tuning stage to optimize a similarity score between document containing the mention and correct entity. However, it raises the issue of ignoring the mention, i.e., it produces identical document vector representation for more than one mentions in a document. Sun et al. (2015) address this by leveraging the semantics of mention, context, and entity. They argue that the representation of a context word is not only influenced by the representation of the words it contains but also by the distance between a context word and a mention. They use CNNs to represent the context as well as to capture the distance between the mention and the context word by encoding the position in continuous space. They use a low-rank neural tensor network to model semantic composition between context and mention as well as entity name and description.

Yamada et al. (2016) constructs an embedding that jointly maps mentions and entities into the same continuous vector space allowing to measure the similarity between any pair of mentions, entities, and mention and an entity. Their overall architecture consists of three models based on the Skip-Gram model each trained with a different objective, namely to predict neighboring words given the target word (Skip-Gram model), to predict the neighborhood of entities given the target entity in a link graph of a KB (KB graph model), and to predict the neighborhood words given the target entity using anchors (e.g. hyperlinks in a Wikipedia article) and their context words (anchor context model). The anchor context model addresses the issue raised if only the KB model and skip-gram model are used, i.e. the vectors of the words and entity can be placed in a different subspace of the vector space. Using their proposed embedding they develop an EL method that computes two types of context, namely textual similarity that is measured according to the similarity between the entity and words in a document in vector space, and coherence that is measured based on the relatedness between the target entity and other entities in a document. The two contexts are combined using a point-wise learning-to-rank algorithm to rank the candidate entities given a mention and a document.

Prior to the application of machine learning techniques in disease normalization, dictionary look-up techniques, and string matching algorithms were used. DNorm (Leaman et al., 2013) was the first machine learning-based disease normalization known to the best of our knowledge. They model the mentions and concepts names as TF-IDF vectors and learn a similarity matrix which determines a similarity score between a pair of TF-IDF vectors. They used pairwise learning to rank specifically a margin ranking loss for training the model. Li et al. (2017) first generated candidates using handcrafted rules, and then use a CNN model to rank the candidates based on the semantic information of the word embeddings. Cho et al. (2017) proposed a method that combines a dictionary and word embedding for normalization. They used cosine similarity between a vector of mention and all the concepts to determine the correct entity in the dictionary.

Our EL method is different from all previous implementation for diseases as we try to exploit taxonomical structure, which has not been explored to best of our knowledge.

2.1.3 Joint Learning

Joint NER and EL has been explored by Sil and Yates (2013) where they leverage a trained NER or chunking system and uses a linking algorithm to decide on a correct candidate among a list of generated mentions candidates. Their model captures the dependency between the linking task and mention boundary decision and re-ranks the candidate mention-entity pairs together. Luo et al. (2015) also jointly optimizes the NER and linking task but they optimize the NER system during training phase unlike Sil and Yates (2013). Thus, their NER system benefits from the entity linking's decision since both the decision are made together. They use a semi-CRF which relaxes the Markov assumption between words in CRF and models the segmentation boundaries directly. Extending further, they also model entity disambiguation and mutual dependency over segmentations. Nguyen et al. (2016) used complex tree-shaped per-sentence CRFs which are derived from dependency parse trees instead of only having connections among mentions and entity candidates. They also use linguistic features from the dependency trees.

In the bio-medical domain, Leaman and Lu (2016) were the first people to jointly model NER and EL together using a semi-Markov model. They use rich features such as parts of speech, 2-,3- and 4-chargrams, etc. for NER, and a supervised semantic indexing (Bai et al., 2010) approach for normalization. Lou et al. (2017) identified that exact inference becomes intractable sometimes if features are defined over long-range dependencies preventing non-local features to be used. To avoid this they use a transition-based model and beam search for designing a system to jointly do disease entity recognition and normalization. All of the models used handcrafted features and task-specific resources, thus cannot leverage semantic or character-level information. Zhao et al. (2018) uses multi-task learning considering NER and EL as a parallel task instead of a hierarchical or sequential task. They use a novel feedback strategy that allows feedback from low-level tasks to high-level tasks and vice versa. They consider both the task of NER and EL as sequence labeling task with same input but a different set of task-dependent tags. Our implementation also uses multi-task learning but the difference is in the underlying task-dependent model, i.e NER and EL. Also, the training procedure is different as in the former model they do a random task selection for training whereas in our case we train both the tasks in parallel.

2.2 Background

In this section we briefly describe few background knowledge, namely Conditional Random Fields, ELMo, *node2vec* and GCN used in rest of the chapters.

2.2.1 Conditional Random Field

In this section, we provide a brief overview of CRFs. For more details one can refer to Sutton and McCallum (2012). The problem of classification is to predict a sequence $\mathbf{y} = (y_0, y_1, \ldots, y_T)$ of random variables given an observed feature sequence $\mathbf{x}$. The elements in $\mathbf{y}$ could have a complex dependency on each other, for example in part-of-speech tagging, each variable y_s is a part-of-speech tag of the word at position s. These complex dependency could be represented by a probabilistic graphical model (PGM). PGMs combine probabilities and independence constraints compactly that allows to factor the representation of a joint distribution over a set of random variables into modular components[1]. This type of learning with graphical models could be approached as generative models that explicitly attempt to model the joint probability distribution $p(\mathbf{x}, \mathbf{y})$ over the input and output. However, because the dimensionality of $\mathbf{x}$ is very large and the features have a complex dependency structure, constructing a probability distribution $p(\mathbf{x})$ over them is hard. Indeed, modeling all the input dependencies leads to intractable models, but without them we get reduced performance. Another approach is using discriminative models, i.e to model the conditional distribution $p(\mathbf{y}|\mathbf{x})$ directly. Such models are known as Conditional Random Fields (CRFs). CRFs combine the advantages of classification and graphical modeling, i.e they enable to compactly model multivariate data by leveraging a large number of input features for prediction. The dependencies that involve only the variable in $\mathbf{x}$ play no role in the conditional model giving a much simpler structure than a joint model. CRFs make independence assumptions among elements of $\mathbf{y}$, and assumptions about how the $\mathbf{y}$ can depend on $\mathbf{x}$, but not among elements of $\mathbf{x}$.

A CRF that follows the first order Markov assumption is called a first order CRF. By relaxing the first-order Markov assumption we can obtain the general CRFs. A linear-chain CRF is defined as follows (Sutton and McCallum, 2012). Let Y, X be random vectors, $\theta = \{\theta_k\} \in \mathbb{R}^K$ be a parameter vector, and

[1]For more details refer to link

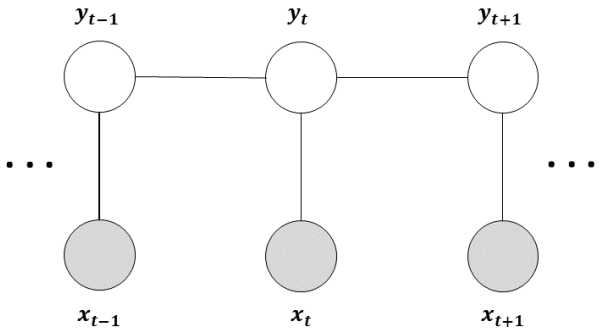


Figure 2.1: Probabilistic Graphical Model for linear-chain Conditional Random Field

$\{f_k(y, \hat{y}, \mathbf{x_t})\}_{k=1}^{K}$ be a set of real-valued features functions. Then a linear-chain conditional random field is a distribution $p(\mathbf{y}|\mathbf{x})$ that takes the form

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x_t}) \right\}, \tag{2.1}$$

where $Z(\mathbf{x})$ is an instance-specific normalization function

$$Z(\mathbf{x}) = \sum_{y} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x_t}) \right\}. \tag{2.2}$$

Feature function are mostly hand engineered indicator functions incorporating the domain knowledge. For instance, in part-of-speech tagging $f(l_i, l_{i-1}, x_i)$ is 0 if l_i is noun and l_{i-1} is verb. Training requires efficient evaluation of the normalizer (eqn. 2.2) which can be done via the forward-backward algorithm. For prediction, Viterbi algorithm, a dynamic programming algorithm, is used to predict the most likely sequence given the sequence of features.

2.2.2 ELMo

Word embeddings such as Word2vec (Mikolov et al., 2013), Glove (Pennington et al., 2014) etc. are widely used in natural language processing. They boost the performance of any neural model significantly. These look-up table embeddings are trained to capture the complex syntactic and semantic characteristic of a word in an unsupervised manner over a large corpus or corpora specific to the domain of the downstream task. This results in a single vector representation for each word, that unfortunately ignores e.g., polysemy or homonymy, i.e., the fact that the meaning of a word is context-dependant and may vary from sentence to sentence. The *deep contextualized* word representation proposed by Peters et al. (2018) addresses this problem. ELMo (Embeddings from Language Models) representations are basically vectors that are assigned to each token in a sentence as a function of the sentence. The vectors are derived from a bidirectional LSTM that is trained with a language model (LM) objective on a large corpus. ELMo uses character convolutions to benefit from subword units such as prefix or suffix and incorporates multi-sense information seamlessly.

Given a sequence of N tokens, $(t_1, t_2, \ldots, t_N)$, a forward LM computes the probability of the sequence by modelling the probability of token t_k given the history $(t_1, \ldots, t_{k-1})$:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k|t_1, t_2, \ldots, t_{k-1}) \tag{2.3}$$

A backward LM is similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token t_k given the future context $(t_{k+1}, ..., t_N)$.

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k|t_{k+1}, t_{k+2}, \ldots, t_N) \tag{2.4}$$

A context-independent token representation $\mathbf{x}_k^{LM}$ is computed using CNN over characters and passed through L layers of forward and backward LSTMs. At each position k, each layer of LSTM outputs a

context-dependent representation $\overrightarrow{\mathbf{h}}_{kj}^{LM}$ and $\overleftarrow{\mathbf{h}}_{kj}^{LM}$, where $j = 1, \ldots, L$, for forward and backward respectively. A bidirectional LM jointly maximizes the log-likelihood of the forward and backward directions.

2.2.3 node2vec

node2vec (Grover and Leskovec, 2016) is a unsupervised method for learning continuous feature representations for vertices/nodes in a graph/network. From here on, we use vertices or nodes interchangeably. A mapping of nodes to a low-dimensional space of features is learned by maximizing the likelihood of preserving the local neighborhoods of nodes. The objective function is defined independent of the downstream task and the representation is learned in a unsupervised way.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a graph where $\mathcal{V}$ is vertices/nodes and $\mathcal{E}$ are edges/links connecting the nodes. Let $f : \mathcal{V} \Rightarrow \mathbb{R}^d$ be the mapping function from nodes to $d-$dimensional feature representation. f is a matrix of size $|\mathcal{V}| \times d$ parameters. For every source node $u \in \mathcal{V}$, $N_s(u) \subset \mathcal{V}$ is defined as the network neighborhood of u generated using a neighborhood sampling strategy S. The objective function is to maximize the log-probability of observing a network neighborhood $N_s(u)$ for a node u conditioned on the feature representation, given by f:

$$\max_f \sum_{u \in \mathcal{V}} \log p(N_s(u)|f(u)) \tag{2.5}$$

The authors make two assumptions to make the optimization problem tractable. First, given the source feature representation, the likelihood of observing a neighborhood node is independent of observing any other neighborhood node:

$$p(N_s(u)|f(u)) = \prod_{n_i \in N_s(u)} p(n_i|f(u)). \tag{2.6}$$

Secondly, the source node and neighborhood node have symmetric effect over each other in the feature space. The conditional likelihood of every source-neighborhood node pair is the softmax of the dot product of the features, measuring how closely the neighborhood representations are in feature space given the feature representation of the source:

$$p(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))} \tag{2.7}$$

Equation 2.5 can be re-written using 2.6 and 2.7 as:

$$\max_f \sum_{u \in \mathcal{V}} \Big[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \Big] \tag{2.8}$$

where $Z_u = \sum_{v \in \mathcal{V}} exp(f(u).f(v))$ and is approximated using negative sampling as it is very expensive to compute.

node2vec uses a flexible sampling strategy, i.e. a biased random walk that can explore the neighborhoods in a breadth-first search or depth-first search way. A random walk of nodes c_i of fixed length l is generated from the following distribution:

$$p(c_i = x|c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if} (v, x) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

where π_{vx} is unnormalized transition probability between nodes v and x, Z is the normalizing constant, $c_0 = u$ is the starting node. Search bias is introduced by setting the unnormalized transition probability to $\pi_{vx} = \alpha_{pq}(t, x).w_{vx}$ where

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{2.10}$$

t is a node at previous time step of the random walk, v is the node at current step, x is a node at next time step, d_{tx} is distance between node t and x, w_{vx} is the weight of the edge between node v and x; w_{vx} defaults to 1 for unweighted graphs. The parameter p controls how likely a node is revisited in a walk and parameter q controls the search for nodes closer to current node or nodes further away.

2.2.4 Graph Convolutional Networks

Graph Convolution Networks (GCNs) are another type of neural models for graph-structured data. A general framework for such networks called Message Passing Neural Networks (MCNN) is defined by Gilmer et al. (2017). As the name suggests, messages are shared among adjacent nodes and thus information flows along the edges in the graph. In a GCN, each layer allows messages only from immediate neighbors, and a stack of multiple layers covers larger neighborhood; k layers implies message is received from nodes at most k hops away.

We use the GCN formulation defined by Kipf and Welling (2016) and should be referred for more details. Let a undirected Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the n vertices/nodes $v_i \in \mathcal{V}$, $\mathcal{E}$ are edges between the nodes and additionally, all nodes are assumed to be connected to itself formulated in the adjacency matrix as $\hat{\mathbf{A}} = \mathbf{I}_n + \mathbf{A}$. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a matrix of d-dimensional features for all n nodes. The layer-wise propagation rule is:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{2.11}$$

where, $\sigma(\cdot)$ is activation functions such as ReLU, $\mathbf{H}^{(l)}$ is the matrix of activation for l^{th} layer, $\mathbf{H}^{(0)} = \mathbf{X}$, $\hat{\mathbf{D}}$ is degree matrix with self-connection, $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$ and $\mathbf{W}^{(l)}$ is a layer-specific trainable weight matrix.

For example, a two-layer GCN can be realized on a graph with adjacency matrix $\mathbf{A}$ by first calculating $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}$. The forward model as a function $f(\mathbf{X}, \mathbf{A})$ is :

$$f(\mathbf{X}, \mathbf{A}) = \sigma(\tilde{\mathbf{A}}\,\mathrm{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)})\mathbf{W}^{(1)}) \tag{2.12}$$

Assumption: The use of spectral graph theory, i.e eigenvectors and eigenvalues of a graph, for clustering nodes matches with the objective of multi-class classification used in Kipf and Welling (2016). In our case, we assume that nodes will learn to have features which help to distinguish a node from its neighboring nodes or nodes having similar semantic or context information as our supervised training objective is to predict the correct node given a feature vector.

Comparison to node2vec: In *node2vec*, the nodes of a graph are translated to sentences by random walks on the graph along the edges from any random nodes. As the number of walks increases, we tend to have node representations that capture all their neighboring nodes withing defined context window. This is similar to GCN in terms defining the number of GCN layers. However, our training objectives are different as *node2vec* is trained in an unsupervised setting whereas GCN is supervised.

Chapter 3

Datasets

Any machine learning application requires high-quality annotated datasets as it plays a crucial role in model performance and its usage. The dataset should be well representative of the problem the algorithm is trying to address. If the dataset is noisy or biased, then the learning algorithm may capture the noise and bias as well (Wei and Dunbrack, 2013; Ashraf et al., 2018). Hence, it becomes a necessity to use an appropriate dataset for addressing a particular problems. Domain-specific datasets curated by experts are less prone to these problems. A prime example from the bio-medical domain is the NCBI disease corpus.

The NCBI disease corpus (Doğan et al., 2014) was prepared by manual annotation by bio-medical domain experts. In it disease mentions were annotated and labelled with their corresponding disease concepts or canonical names from a total of 793 PubMed [1] abstracts. PubMed is a free search engine accessing primarily the MEDLINE database (a database of life science and bio-medical information) of references and abstracts on life sciences and bio-medical topics[2]. A *mention* or *entity* is a word or a span of words which of interest, like disease names. Each mention is mapped to standard disease controlled vocabulary *concepts* (canonical names), namely MeSH [3] and OMIM [4] terms.

In section 3.1 we discuss in detail the NCBI dataset followed by section 3.2 on the MeSH taxonomy. We make explicit our assumptions and the limitations of using this dataset for addressing disease normalization in section 3.3.

3.1 NCBI disease corpus

The National Center for Biotechnology Information (NCBI) disease corpus (Doğan et al., 2014) is a well-known disease corpus in the bio-medical community. The NCBI disease corpus is better than other existing disease corpora because of the following reasons: 1) All the sentences in the abstracts are annotated which is useful for higher-level text mining task that explores relationships between entities within the abstract. 2) The corpus is considered to be several times larger than other manually annotated disease corpus to best of our knowledge. 3) The annotations are mapped to MEDIC (Davis et al., 2012) vocabulary for assigning disease concepts/canonical names. 4) The annotations are performed by domain experts with each annotation completed by at least two individuals.

All the annotations in the disease corpus are completed following certain annotations rules, namely, 1) A concept is annotated that matches the preferred name (canonical name). 2) A concept is annotated that matches a synonym of the preferred name/concept unless there is another concept that matches the preferred name. 3) The most specific concept is annotated that correctly describes the disease mention. 4) The closest hypernym concept is annotated that logically describes the disease mention. 5) *Composite disease mentions* are annotated using the "|" separator. 6) Multiple concepts are annotated using "+" concatenator to logically describe the disease mentions. 7) Gene names are also annotated as disease mentions if that mention is used interchangeably. 8) Specific concepts in OMIM that are not included in MEDIC are used whenever necessary.

[1] https://www.nlm.nih.gov/databases/download/pubmed_medline.html
[2] from Wikipedia
[3] https://meshb.nlm.nih.gov/search
[4] https://www.omim.org/

Corpus characteristics	Training set	Validation set	Test set	Whole Corpus
PubMed citations	592*	100	100	792*
Total disease mentions	5134	787	960	6881
Unique disease mentions	1691	363	424	2136
Unique concept IDs	657	173	201	790
Concepts to mentions	2.55	2.09	2.10	2.74

Table 3.1: NCBI disease corpus for disease name recognition. *We dropped a repeating abstract

10021369 | t | Identification of APC2, a homologue of the adenomatous polyposis coli tumour suppressor.
10021369 43 76 adenomatous polyposis coli tumour Modifier D011125

This is a sample from the dataset where *adenomatous polyposis coli tumour* is a disease mentioned in the title of the abstract and is mapped to MeSH identifier(ID) D011125. MeSH is explained in the next section 3.2.

The disease mentions are categorized in four categories: *specific disease*, *disease class*, *composite mention* and *modifier*. However, we ignore in this thesis specific categorizations and consider only the disease level for entity identification. The disease corpus has three splits namely training, validation and test set (table 3.1) and consist of 593 [5], 100 and 100 abstracts respectively. There are 5.08 disease mentions and 3.28 disease concepts per abstract. There are 5134, 787 and 960 disease mentions respectively in training, validation, and test set with 1691, 363 and 424 unique disease mentions respectively. There are 657, 173 and 201 unique concept IDs in each of the data splits respectively. The disease concepts are on average mapped to as 2.55, 2.09 and 2.10 disease names respectively.

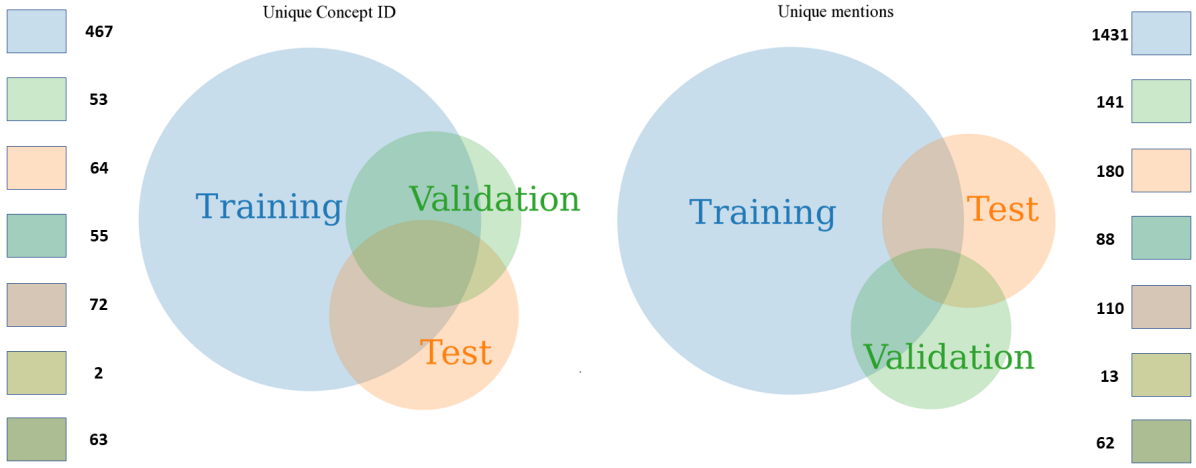


Figure 3.1: Venn diagram of unique concept IDs and unique mentions in training, validation and test set.

From figure 3.1, the number of unique concepts and unique mentions common to both training and test set are 135 and 172 respectively, whereas concepts and mentions outside the training set are 66 and 193 respectively. Note that it is valid to assume that the model has generalization ability if it predicts a mention that has never appeared in the training set but whose mapped concept did appear there. This is because every concept may be referred to by more than one name.

Set	MeSH(unique)	OMIM(unique)	Annotated Concepts (unique)
Training	1512(599)	198(71)	1710(670)
Validation	316(153)	52(23)	368(176)
Testing	378(182)	49(21)	427(203)

Table 3.2: Distribution of MeSH and OMIM concepts

[5]Document_8528200 is repeated, hence we use only 592

The disease vocabulary used for mapping mentions to concepts is MEDIC (MErged DIsease voCabulary). It consists of the "disease" branch of MeSH, with OMIM identifiers for genetic disease. We briefly describe the MeSH in the next section.

3.1.1 Preprocessing

We convert the dataset into the Brat standoff format[6] which is one of the standard annotation formats used in the (bio)NLP community. This allows us to use visualization tools such as the Brat annotation tool[7] and Elsevier in-house tools and packages. We use the NLTK[8] sentence and word tokenizer for preprocessing the data.

3.2 MeSH Taxonomy

The *Medical Subject Heading* (MeSH ®) thesaurus is a bio-medical controlled vocabulary produced by National Library of Medicine[9]. It is used for indexing, cataloging and searching bio-medical and health-related documents and information. MeSH records can be classified into three classes, namely *Descriptors*, *Qualifiers* and *Supplementary Concepts records*. Descriptors are mainly used for indexing and retrieval. Descriptors are organized into a numbered tree structure or hierarchy from semantically broader concepts and topics to narrower ones.

MeSH descriptors are organized in 16 categories, for instance, category A refers to anatomic terms, category B for the organism, category C for disease, etc. Each category is further divided into subcategories where descriptors are arrayed hierarchically from most generic to most specific in up to thirteen hierarchical levels. Though it is referred to as MeSH *tree*, because common nodes are repeated multiple-times, it is not a naive tree (where siblings have a unique ancestor), but forms in reality a directed acyclic graph. We consider only the disease branch for our use.

Figure 3.2: This is a disease concept with preferred name Adenoma and unique ID D000236. Entry terms are synonyms and Scope Note is a description of the disease. The position in the MeSH taxonomy is given by the Tree Number. A concept can be repeated at multiple tree node in the MeSH tree each with different tree number. Taken from https://meshb.nlm.nih.gov/record/ui?ui=D000236 on July 9th, 2019

A descriptor node has many attributes of which the following are of key interest, namely, *MeSH*

[6] http://brat.nlplab.org/standoff.html
[7] http://brat.nlplab.org/index.html
[8] http://www.nltk.org/
[9] https://www.ncbi.nlm.nih.gov/mesh

Heading, *Tree Number(s)*, *Unique ID*, *Scope Note* and *Entry Term(s)*[10] (figure 3.2). The MeSH heading is the term used in the MEDLINE[11] database as the indexing term. Tree numbers indicate the places where the MeSH heading appears and are the formal computable representation of the hierarchical relationships. A unique ID is an identifier that is allocated to each descriptor to uniquely identify it among all others. A scope note is a short piece of text which captures the meaning of the MeSH term. Entry terms are synonyms, alternate forms, and other closely related terms that are generally used interchangeably with the preferred term.[12]

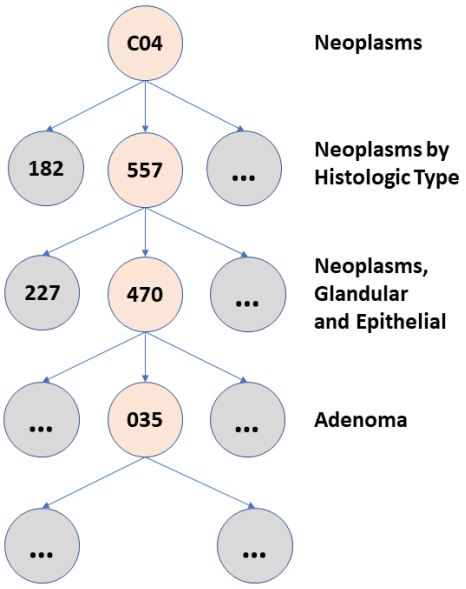


Figure 3.3: A Tree number is used to identify the position of the disease in the MeSH taxonomy. For the concept D000236, the tree number is C04.557.470.035. Each number separated using period as delimiter denotes a node in the tree. A node can have multiple children.

Each descriptor may appear in more than one subcategories as may be appropriate. A tree number is assigned to each descriptor depending on the location in the tree, and maybe followed by one or more additional numbers (figure 3.3). The number serves only to locate the descriptors in the tree and to alphabetize those at a given tree level, but doesn't have any intrinsic significance. They are subjected to change when new descriptors are added or the hierarchical arrangement is revised to reflect the changes in vocabulary.

3.3 Assumptions and Limitations

This dataset contains around 76 normalized concepts that are composite or multiple disease mentions. During training, we choose at random any one of the concepts to be mapped to a mention. By random sampling, we are assuming that in high-dimensional space these concepts are close to one another and individually can represent the disease mention. At test time, we assume that selecting any one of the multiple concepts is correct.

As we already mentioned, the vocabulary or the KB used for mapping the mentions is the disease branch of MeSH, with OMIM identifiers. Since we address the linking problem to leverage information from the taxonomy, we limit ourselves to only those diseases which are part of the "disease" branch of MeSH. Any concept which is not part of the "disease" branch is either converted to a MeSH *descriptor* or ignored from normalization. Specifically, we convert OMIM concepts to their related MeSH concept using the Comparative Toxicogenomics Database [13]. If the related MeSH concept is a *supplementary concept records*, we convert them to *descriptor* MeSH using a script[14]. After filtering, we skip a total of 192

[10] Another example: https://meshb.nlm.nih.gov/record/ui?ui=D011125
[11] https://www.nlm.nih.gov/bsd/medline.html
[12] for more details: https://www.nlm.nih.gov/mesh/meshrels.html
[13] http://ctdbase.org/downloads/;jsessionid=644F320D212CF859C2A7FB061C9D9716#alldiseases
[14] provided in the supplementary material

occurrences of 33 unique concepts in the training set, 31 occurrences of 12 unique concepts in validation and 33 occurrences of 12 unique concepts on the test set. Thus for normalization, we consider 4942, 756 and 927 mentions in training, validation, and test set respectively.

The MeSH tree contains diseases that are replicated across multiple nodes. To have a proper semantic representation of a disease we need to use the information of its neighborhood. We do this by converting the tree into a graph with nodes having multiple incoming and/or outgoing nodes. We also ignore the directions and use an undirected graph. The average degree of a node is 3 with a maximum degree of 63 in a total of 4818 disease nodes. This allows us to learn a representation of disease that uses information from multiple branches.

3.4 Other details

Implementation details: We implemented all our models using PyTorch framework and trained on a system with 32 GB RAM and NVIDIA 1080 with 8 GB graphic memory.

Supplementary material: All supplementary materials are made available at Google Drive

Chapter 4

Disease Recognition

NER is typically modeled as a sequence labelling problem which can be formally defined as follows: Given a sequence of input tokens $\mathbf{x} = (x_1, x_2, ..., x_n)$ and a set of labels $\mathbf{L}$, determine a sequence of labels $\mathbf{y} = (y_1.y_2, ..., y_n)$ such that $y_i \in \mathbf{L}$ for $1 \leq i \leq n$. The label incorporates two concepts: 1) the position of the token within the entity by using **BIO** tagging scheme (Ramshaw and Marcus, 1995) and 2) the type of the entity e.g. name of a person, location or disease.

Each token is assigned one of the three labels denoting **B**eginning of an entity, **I**nside an entity or **O**utside in **BIO** tags scheme. For example, **B_xxx**, **I_xxx** and **O** where **xxx** denotes the type of tag.

$$\underbrace{\textit{An}}_{\text{O}}\ \underbrace{\textit{evaluation}}_{\text{O}}\ \underbrace{\textit{of}}_{\text{O}}\ \underbrace{\textit{genetic}}_{\text{O}}\ \underbrace{\textit{heterogeneity}}_{\text{O}}\ \underbrace{\textit{in}}_{\text{O}}\ \underbrace{\textit{145}}_{\text{O}}\ \underbrace{\textit{breast-ovarian}}_{\text{B\_Disease}}\ \underbrace{\textit{cancer}}_{\text{I\_Disease}}\ \underbrace{\textit{families.}}_{\text{O}}$$

Another tagging scheme used in NER is **IOBES** with two new tags namely, **E**nd of entity and **S**ingle entity (Ratinov and Roth, 2009). This tagging scheme is found to provide greater discriminative power to machine learning algorithms and hence better performing in some cases. We use the former tagging scheme for NER.

In the next section, we describe the network architecture for this task followed by section 4.2 on evaluation, results and analysis. In section 4.3, we discuss our observations and ways to improve our results further.

4.1 Method

The baseline models we consider for disease NER are by Lample et al. (2016) and by Ma and Hovy (2016). The basic architecture is as follows (figure 4.1):

Given a sentence, $\mathbf{x}_{1:n} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i$ is a word embedding, a forward LSTM (Hochreiter and Schmidhuber, 1997) $\overrightarrow{f}$ encodes the contextual information from left to right into a d-dimensional vector $\overrightarrow{\mathbf{h}_t}$ for each position t. Similarly, a backward LSTM $\overleftarrow{f}$ is used to encode each $\mathbf{x}_t$ from right to left into $\overleftarrow{\mathbf{h}_t}$. Both vectors, $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$, are concatenated to represent a vector for $\mathbf{x}_t$ that captures the long distance dependencies in the given sequence.

$$\begin{aligned} \overrightarrow{f}(\mathbf{x}_t) &= \overrightarrow{\mathbf{h}_t} \qquad \text{and} \qquad \overleftarrow{f}(\mathbf{x}_t) = \overleftarrow{\mathbf{h}_t} \\ f(\mathbf{x}_t) &= [\overrightarrow{f}(\mathbf{x}_t); \overleftarrow{f}(\mathbf{x}_t)] = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}] = \mathbf{h}_t \end{aligned} \tag{4.1}$$

Each vector $\mathbf{x}_t$ represents a word in the sentence which is a concatenation of pre-trained word embedding using skip-gram, and a character-level embedding trained either using a LSTM Lample et al. (2016) or CNN layer Ma and Hovy (2016). The dimensionality of the encoded vectors $\mathbf{h}_t$ is reduced to $\mathbb{R}^k$, where k is a number of distinct tags, using a linear transformation. The reduced vectors (or equivalently we can call them as score values) for each tag are then used as input to the CRF layer (section 2.2.1) to model the tags jointly instead of modeling the decision independently. We consider the CRF scoring output in the form of a matrix $\mathbf{P}$ of size $n \times k$ where $P_{i,j}$ corresponds to the score of j^{th} tag of the i^{th} word in a sentence. For a sequence of predictions, $\mathbf{y}_{1:n} = (y_1, y_2, \ldots, y_n)$ the CRF score is defined

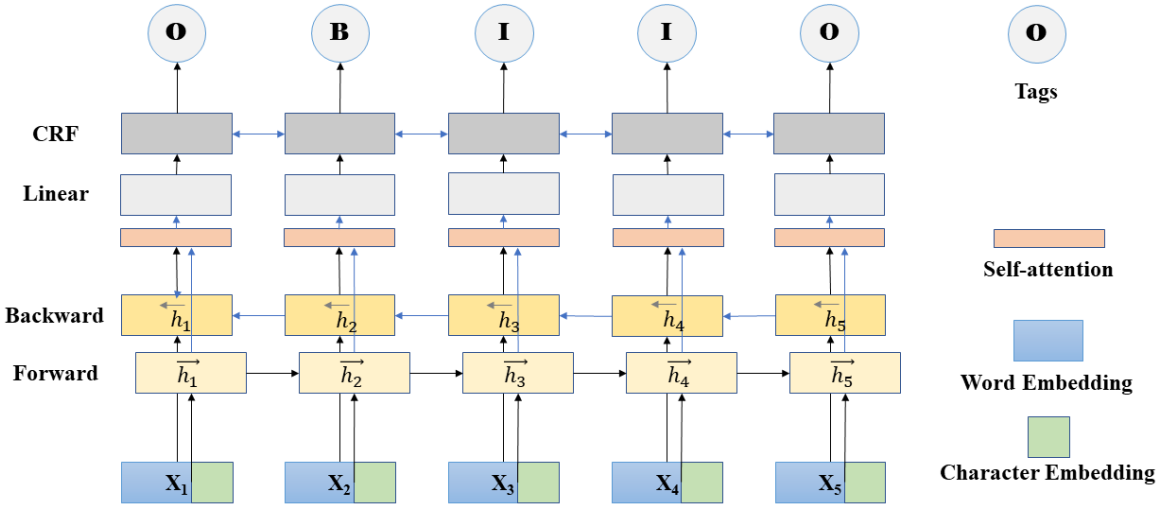


Figure 4.1: **NER model architecture**: A bidirectional LSTM layer followed by a self-attention and linear transformation, whose outputs are forwarded to a CRF layer. Each $\mathbf{X}_i$ represents the word embedding for the word at the i^{th} , which can be concatenation of pre-trained word or ELMo embeddings, and character embeddings.

as

$$s(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \tag{4.2}$$

where $A_{i,j}$ represents the score of a transition from tag i to tag j forming a matrix $\mathbf{A}$ of learned transition scores. A corresponds to the transition probability and P to the emission probability matrix of traditional linear chain CRFs. y_0 and y_n are also included, denoting the start and end tags of a sentence. $\mathbf{A}$ is thus a square matrix of size $k + 2$.

Assuming a Gibbs distribution over the possible tag sequence in eqn. 4.3 we maximize the log-probability of the correct tag sequence (eqn. 4.4) as our training objective.

$$p(\mathbf{y}_{1:n}|\mathbf{x}_{i:n}) = \frac{\exp\left(s(\mathbf{x}_{1:n}, \mathbf{y}_{1:n})\right)}{\sum_{\hat{y}_{1:n} \in \mathbf{Y}_{(\mathbf{x})}} \exp\left(s(\mathbf{x}_{1:n}, \hat{y}_{1:n})\right)} \tag{4.3}$$

$$\log(p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n})) = s(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) - \log\left(\sum_{\hat{y}_{1:n} \in \mathbf{Y}_{(\mathbf{x})}} \exp\left(s(\mathbf{x}_{1:n}, \hat{y}_{1:n})\right)\right) \tag{4.4}$$

where $\mathbf{Y}_{(\mathbf{x})}$ represents all possible tag sequences for a sentence $\mathbf{x}_{1:n}$. Due to the Markov assumption in equation 4.2, the forward-backward evaluates the denominator of equation 4.3 efficiently. At prediction time, the output sequence of all the possible sequences $\mathbf{Y}_{(\mathbf{x})}$ is the one that obtains the maximum score, using Viterbi decoding, given by:

$$\mathbf{y}^*_{1:n} = \underset{\hat{y}_{i:n} \in \mathbf{Y}_{(\mathbf{x})}}{\arg\max}\ s(\mathbf{x}_{1:n}, \hat{y}_{1:n}) \tag{4.5}$$

We modified the base model with a number of changes that enhance the performance of the task and interpretability of the model. The changes are as follows:

Self-attention: In the case of medical documents, a lot of the disease names are abbreviated. These abbreviations are often decided by authors of the respective documents. The problem is that a pre-trained embedding might not be sufficient enough to correctly represent the meaning to make it identifiable as an entity. Another scenario would be when these abbreviations do not appear in the embedding's

training corpus, although lack of word embedding coverage can be to some extent mitigated by the use of a character embedding. To address this, we used self-attention (Vaswani et al., 2017) to allow other words in the sentence or the document to contribute to the representation of the abbreviated words. The self-attention mechanism is applied after the BiLSTM layers.

Let $\mathbf{H}$ represent all the outputs of BiLSTM layers, $\mathbf{h}_t$ for each word in a sentence in matrix form; $\mathbf{H} \in \mathbb{R}^{n \times 2d}$ where n is the number of words in a sentence and d is dimension of $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ each. We evaluate the output in form of matrix $\hat{\mathbf{H}}$ as follows:

$$\hat{\mathbf{H}} = \text{softmax}(\mathbf{H}\mathbf{H}^T)\mathbf{H} \tag{4.6}$$

where $(\mathbf{H}\mathbf{H}^T) \in \mathbb{R}^{n \times n}$ is matrix of weight of each word on every other word. The new representation $\hat{\mathbf{H}}$ is then used to evaluate the scores by linear transform followed by the CRF layer.

Character Attention: We use an attention mechanism to dynamically decide how much weight should be given to the word-level or character-level component instead of direct, naive concatenation as proposed by (Rei et al., 2016). Let $\mathbf{h}_i^w \in \mathbb{R}^d$ be a word embedding and $\mathbf{h}_i^c \in \mathbb{R}^e$ be character embedding for word at i^{th} position. We expand the dimension of the character embedding to d dimensions using a linear transform (a dense layer). We differ here from the original implementation by the author in which they introduce a $\tanh$ activation following the linear transform.

$$\mathbf{m} = \mathbf{W}_c \mathbf{h}_i^c \tag{4.7}$$

Vector $\mathbf{z} \in \mathbb{R}^d$ is used to determine the weights dynamically which is then used to obtain the new word representation $\hat{\mathbf{h}}_w$

$$\begin{aligned} \mathbf{z} &= \sigma(\mathbf{W}_z^{(3)} \tanh\left(\mathbf{W}_z^{(1)}\mathbf{h}_w + \mathbf{W}_z^{(2)}\mathbf{m}\right)) \\ \hat{\mathbf{h}}_w &= \mathbf{z}\mathbf{h}_w + (\mathbf{1} - \mathbf{z})\mathbf{m} \end{aligned} \tag{4.8}$$

where $\mathbf{W}_z^{(1)}$, $\mathbf{W}_z^{(2)}$ and $\mathbf{W}_z^{(3)}$ are weight matrices for calculating $\mathbf{z}$ and $\sigma()$ is logistic function.

ELMo embedding: We use an ELMo embedding (section 2.2.2) to represent the words instead of pre-trained word embeddings and character-level embeddings. These embeddings are a deep contextual representations of words in a sentence, i.e. the embedding is now a function of its sentential context rather than a simple dictionary look-up table of pre-trained embeddings.

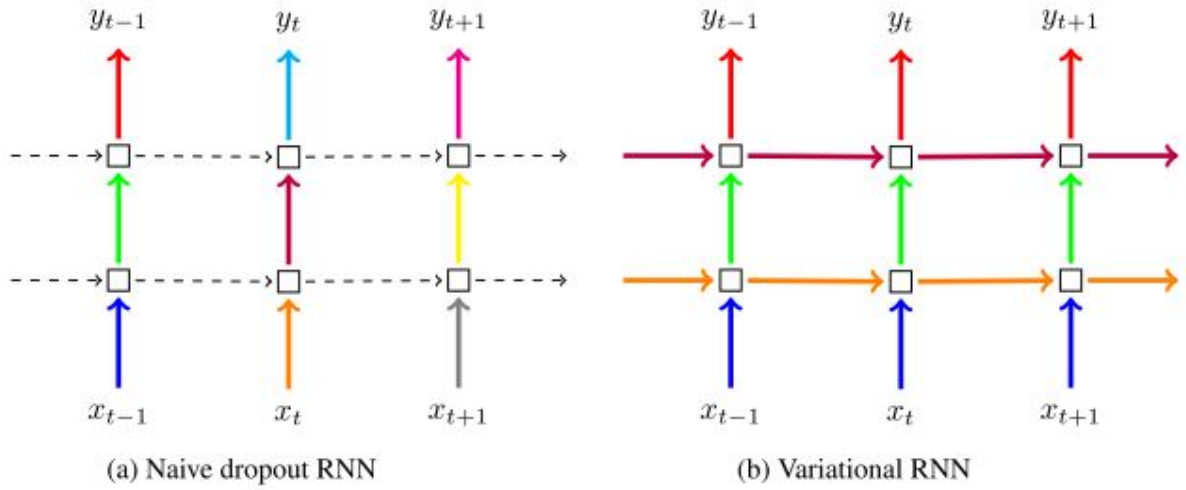


Figure 4.2: **Variational Dropout**: Each square box represents an RNN unit with the horizontal arrow representing recurrent connections. Naive dropout (left) drops only input and output connections at random. Variational dropout (right) uses the same dropout mask at each time step. (Image from (Gal and Ghahramani, 2016))

Variational Dropout: Dropout is used to regularize deep neural networks which are very powerful function approximators. During training, networks units are randomly masked (dropped) but this technique has not been successfully used in RNNs where it is believed to add instead noise to the recurrent layers. Variational Dropout in RNNs (Gal and Ghahramani, 2016) is a technique where the same dropout mask is repeated at each time step for both inputs, outputs and recurrent layers (figure 4.2).

4.2 Result and Analysis

Evaluation metrics: The standard metrics used for evaluating any sequence labelling task are *precision*, *recall* and *F1-score*. Let **tp** be the number of labels of a class that are predicted correctly, **fp** be the number of labels of a class wrongly predicted and **fn** be the number of labels that predict a class but not the correct one. Precision is defined as

$$Precision = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{4.9}$$

, recall as

$$Recall = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{4.10}$$

and F1-score as

$$\textit{F1-score} = 2 \times \frac{\textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}. \tag{4.11}$$

In NER, the output are **BIO** tags with class type information. We combine the **B** and **I** tags to identify exact multi-word disease names which implies we only have binary tag that is either a disease or non-disease. We evaluate on these combined tags with respect to the gold set and report out scores. For example, $\underbrace{\textit{breast-ovarian}}_{\text{B\_Disease}}\ \underbrace{\textit{cancer}}_{\text{I\_Disease}}$ is given a tag *Disease* where as $\underbrace{\textit{families.}}_{\text{O}}$ is a *Non-Disease*.

Parameter	Default Values
Char embedding (LSTM)	60
Char embedding (CNN)	60
Word embedding (Word2vec)	200
Word embedding (ELMo)	1024
Word RNN units	100
Dropout	0.5
Self-attention	✓
Sentence split	✓
Character attention	✗

Table 4.1: Default values of parameter for all the experiment unless specified.

We select our models using early stopping based on the F1-score after each epoch on the validation set for each of the run in an experiment. We then report the mean and standard deviation of our results on the test set by evaluating on each run in an experiment.

Training details: For all our experiments, we train our network using back-propagation algorithm and update our parameter using *stochastic gradient descent* (SGD) with *ADAM* (Kingma and Ba, 2014) optimizer, *learning rate* of 0.001, *batch size* of 32 and run for 100 *epochs*. We use the following default setting unless specified in the experiment. We split our abstracts into sentences using nltk[1] and use one sentence at a time (table 4.1). Our *word embedding dimensions* for pre-trained word2vec embedding is 200 and 1024 for ELMO. For character embedding, we use the last output of LSTM as our representation. Thus, the *char embedding dimensions* is 60 from LSTM with hidden units of 30. For our CNN, we deviate from the original and standard implementation of character embedding which usually comprises a single layer of CNN with different kernel size followed by a max-pooling layer over time. Instead, we use two stacked CNN layers each of kernel size 3, which is similar to having a kernel of size 5 In doing so, we add more flexibility to the model to decide the kernel size parameters by itself and also the number

[1] https://www.nltk.org/

of overall parameters is reduced. We use max-pooling of kernel size 3 between the CNN layers and max pool over time at the final output. Our hidden units in *word rnn* is 100 for each forward and backward LSTM. We use a self-attention mechanism by default in all our experiments and naive *dropout* of 0.5 for all the LSTM layer unless specified.

Model	Parameter	Value	Precision ↑	Recall ↑	F1-score ↑
Lample et al. model					
Lample et al.* (Baseline 1)	self-attention	✗	0.824 ± 0.022	0.742 ± 0.019	0.781 ± 0.003
Baseline 1 +	-	Default	0.796 ± 0.017	0.756 ± 0.007	0.775 ± 0.006
Baseline 1 +	sentence split	✗	0.763 ± 0.036	0.687 ± 0.017	0.722 ± 0.012
Baseline 1 +	word RNN units	150	0.812 ± 0.017	0.759 ± 0.012	0.785 ± 0.010
Baseline 1 +	dropout	0.3	0.790 ± 0.028	0.748 ± 0.015	0.768 ± 0.016
Baseline 1 +	dropout	0.1	0.826 ± 0.014	0.741 ± 0.012	0.781 ± 0.009
Baseline 1 +	character attention	✓	0.831 ± 0.012	0.760 ± 0.017	0.794 ± 0.007
Ma and Hovy model					
Ma and Hovy* (Baseline 2)	self-attention	✗	0.823 ± 0.011	0.776 ± 0.023	0.799 ± 0.012
Baseline 2 +	-	Default	0.827 ± 0.020	0.781 ± 0.005	0.803 ± 0.008
Baseline 2 +	sentence split	✗	0.751 ± 0.023	0.721 ± 0.023	0.735 ± 0.008
Baseline 2 +	word RNN units	150	0.828 ± 0.013	0.782 ± 0.016	0.804 ± 0.004
Baseline 2 +	dropout	0.3	0.804 ± 0.027	0.786 ± 0.018	0.794 ± 0.013
Baseline 2 +	dropout	0.1	0.828 ± 0.020	0.784 ± 0.008	0.806 ± 0.007
Baseline 2 +	character attention	✓	0.839 ± 0.020	0.780 ± 0.018	0.808 ± 0.006
ELMo embedding					
Baseline +	word embedding	ELMo	0.854 ± 0.008	0.873 ± 0.005	0.863 ± 0.004
ELMO embedding + Variational Dropout					
Baseline ++	Dropout	0.5	0.878 ± 0.003	0.856 ± 0.005	0.867 ± 0.002

Table 4.2: Results on test set of the experiments with different parameter settings and modification on baseline models We use Lample et al. (2016) and Ma and Hovy (2016) models as a baseline. For ELMo we use the same architecture common to both the baseline. All other parameters if not specified are set to default. Results reported are the mean and standard deviation of 5 runs. *our implementation

4.2.1 Quantitative Analysis

In table 4.2, we report our results on test set of our experiments on our modified models and compare with our baseline models . Note that the default setting for our tests is not that of our baseline models, and we explicitly mention which parameters we changed. We see an improvement in F1-score in all the experiments when using character CNN embedding over LSTM. Corpus pre-processing, viz., splitting into and inputting the corpus as sentences rather than as complete abstracts also improves a lot. Using a naive dropout of 0.1 is little better than 0.3 or 0.5. Using self-attention shows a slight drop in performance, however we use it because it provides model interpretability. The model also gains some improvement if we increase the number of hidden units. Our best models in both baselines are those with character attention. Introducing ELMo embeddings with a default parameter setting improves the model performance by ≈ 0.06 F1-score over our best model. Regularizing our ELMo model with recurrent dropout improved the model precision but effects the recall keeping the F1-score almost the same.

4.2.2 Qualitative Analysis

Our best model, ELMo with variational dropout improves over baseline models in predicting single abbreviated disease names, such as *T-PLL*. In certain cases, variations of disease names recognized earlier in the abstract are not recognized, for example, *sporadic T-PLL*. Certain disease names are predicted with

additional leading or trailing words, for example *unilateral retinoblastoma* is predicted as *isolated unilateral retinoblastoma* with leading word *isolated* by all models. The reason for this could be the ability of the model to identify noun phrases as disease names. Our baselines mark or mistake *deficiency* or *apparent deficiency* as a disease, whereas ELMO models is able to distinguish meaning of deficiency correctly. Based on the type of correct predictions and errors made by our model with ELMo embeddings, we can conclude that deep contextualized embedding does help in disease NER.

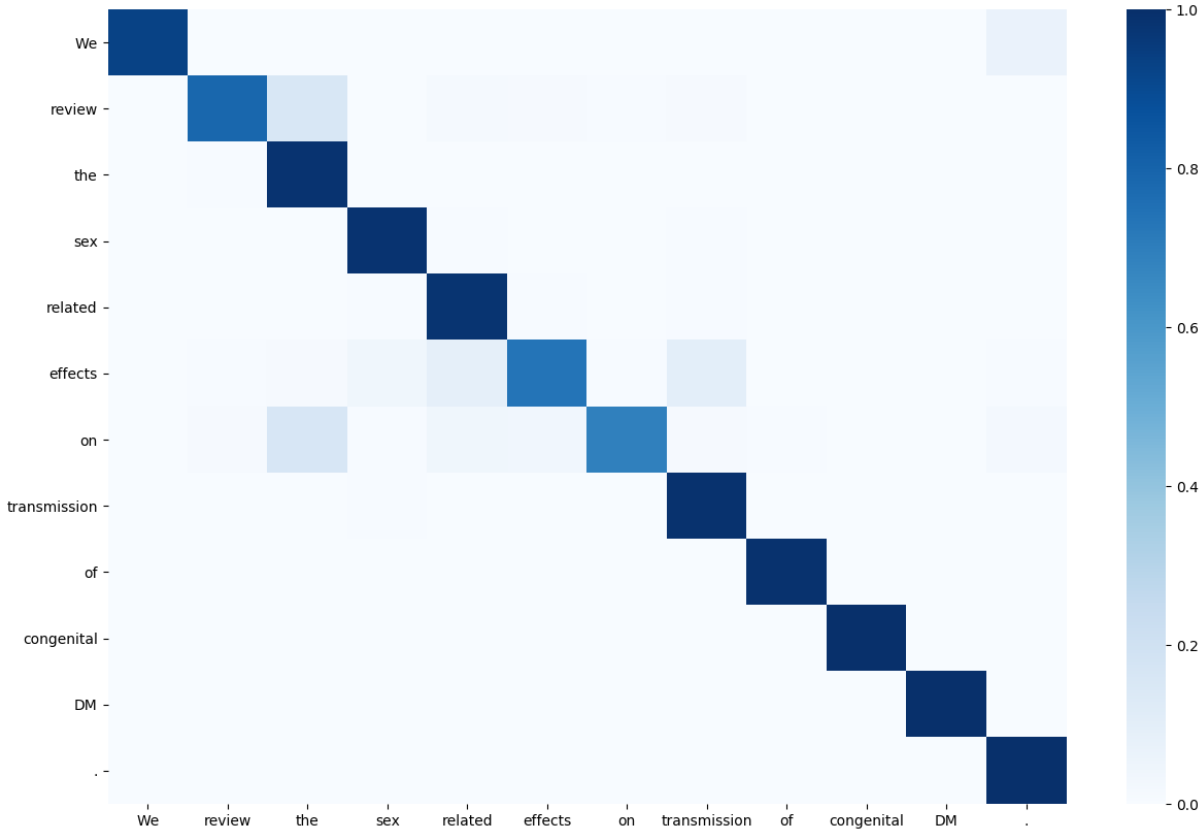
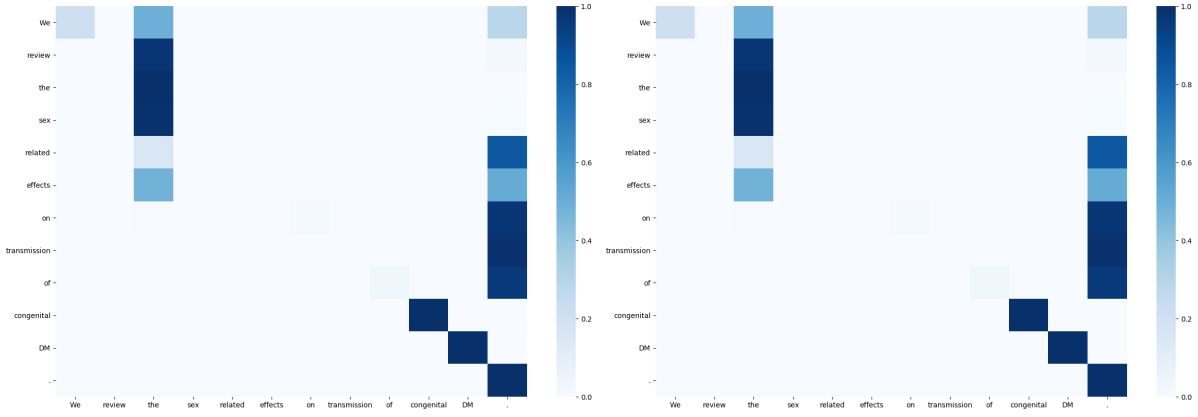


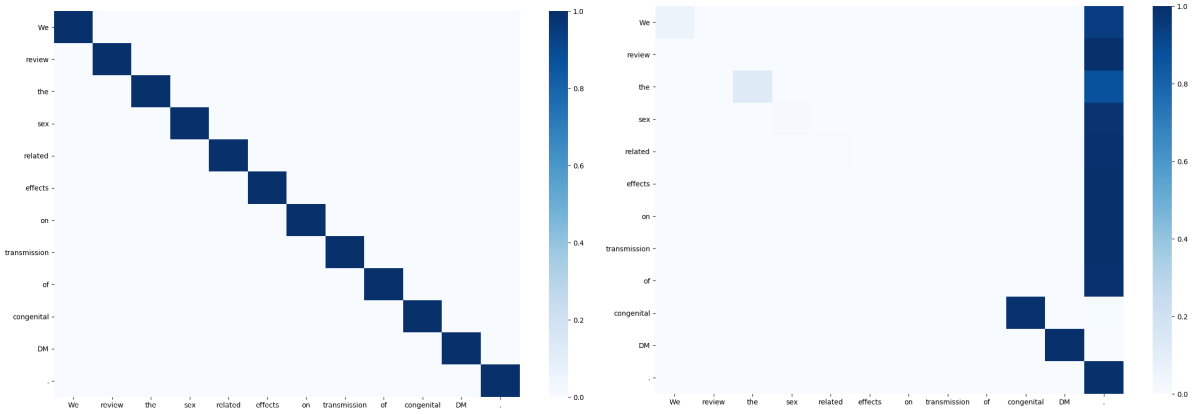
Figure 4.3: Attention weight of word w.r.t. all other words using the self-attention mechanism over ELMo with variational dropout model. Horizontal rows of values for each word sum up to one. Words have highest similarity with themselves as evident from the diagonal.

In figure 4.3, we plot a heatmap of the self-attention mechanism of ELMo with variational dropout model. We see that *congenital DM* is a disease name which shares negligible information with its neighboring words. This invalidates our assumption that self-attention will help abbreviated words and entities such as *DM* to become more informative by fetching information from its neighboring words. However, in case of the word *effects* it leverages information from words such as *sex, related and transmission*. It is also observed that prepositions, conjunctions, and article show a similar behavior w.r.t. the self-attention mechanism. Analyzing further the behavior of self-attention on models trained with Skip-Gram word embeddings and LSTM based character embeddings in figure 4.4, we observe that attention weights are mostly concentrated on stop words: articles, punctuation marks, prepositions, etc. Re-weighting between the word and character embedding does not change anything. Also, from figure 4.5 we observe that using the CNN based character embedding and concatenating it with a word embedding results in "reflexive" attention weights focused only the current word. However, re-weighting word and character embedding does shift the weights for punctuation mark, but not for disease names. This shows that the model emphasizes word embedding over character attention for disease names and on character embedding for every other word. The difference between the LSTM and CNN based character embedding is that the weight distribution for the LSTM embedding falls mostly on articles, prepositions, and punctuation. For the CNN embedding it falls on the other hand mostly on punctuation, specifically on periods. In the absence of a period at end of a sentence, the weight is distributed mostly along the diagonal. This is usually the behaviour in case of using self-attention on top of recurrent encodings.



(a) self-attention (b) character attention

Figure 4.4: Attention weight of a word w.r.t. to all other words in the self-attention mechanism of the Skip-Gram word embedding and LSTM-based character embedding. Weights are mostly allocated on the article or punctuation mark except for disease names. Character attention (right) is not really different to word-character embedding concatenation (left). We use the same sentence as figure 4.3.



(a) self attention (b) character attention

Figure 4.5: Attention weight of a word w.r.t. all other words in self-attention mechanism of the Skip-Gram word embedding and CNN-based character embedding. Using concatenation of word and character embeddings (left) results in attention concentrating across the diagonal (words are similar only to themselves). However, re-weighting word and character embeddings (right) with attention gives clear indication that words which are not disease names can have similar embeddings, and hence all words except disease names are most similar to period. We use the same sentence as figure 4.3.

4.3 Discussion

In table 4.3, we give few examples of disease name recognition by different models, namely Lample et al. (2016) (baseline 1), Ma and Hovy (2016) (baseline 2) and ELMo with variational dropout (baseline++). We observe that ELMo model is able to identify the abbreviated disease names much better than our baselines. This might be because the ELMo starts at character-level and gradually captures the sentence-level semantic information. For example, T-PLL is identified in all the text. However, in case of sporadic T-PLL only T-PLL portion is identified as disease by ELMo model ignoring the sporadic word as part of disease name. Similarly, in sporadic T-cell prolymphocytic leukaemia disease name, ELMo model ignores sporadic as part the name. This could be because of ill representation of contextual embedding of sporadic.

In certain case, disease names are partially identified, for example, in B-cell non-Hodgkins lym-

Disease name	Baseline 1	Baseline 2	ELMo
T-PLL	✗	✓**	✓
sporadic T-cell prolymphocytic leukaemia	✓	✓	✗*
sporadic T-PLL	✗	✗	✗*
B-cell non-Hodgkins lymphomas	✗*	✗*	✓
B-NHL	✓**	✓**	✓
unilateral retinoblastoma	✗*	✗*	✗*
C5D	✓**	✓**	✓**
sporadic prostate cancer	✓**	✓	✓
SCA1	✗	✗	✓
SCA2	✗	✗	✓
SCA3	✓	✗	✓

Table 4.3: A few examples of disease names recognition by different models, namely Lample et al. (2016) (baseline 1), Ma and Hovy (2016) (baseline 2) and ELMo (baseline++). '✓' denotes disease is identified exactly and '✗' denotes disease is not identified to correct span. '*' with '✗' denotes parts of disease name is identified and '**' with '✓' denotes disease is not always exactly identified within a abstract.

phomas, both the baseline model ignores B-cell as part of disease names whereas ELMo model identifies B-cell correctly. Here, contextual embedding turns out to be useful as compared to previous example.

The three example of disease name, namely SCA1, SCA2 and SCA3 are identified by ELMo model correctly as compared to baselines which support our claim that ELMo model identifies abbreviated disease names better than other models.

An observation worth noting is that the models sometimes fail to identify repeated occurrences of the same (or similar) disease names within the same abstract. A post-processing step to mark re-occurring mentions with the same disease tag within an abstract would increase the overall recognition of diseases.

Similarly, considering that different models predict different span of words as disease names, using ensemble models with a majority voting scheme before combining **BIO** tags would further boost the F1-score. This is because different models might identify some disease correctly in **BIO** level, like identifying the beginning correctly or subsequent inside tags. By taking majority voting we select those BIO tags which is agreed upon by most models thus identifying the disease span correctly.

Chapter 5

Disease Normalization

A crucial step in textual enrichment is the task of *normalizing* named entities detected in text. This task is also known as *Entity Linking* (EL) and consists in disambiguating – linking – an entity against a *canonical* identifier derived from a manually curated knowledge repository, such as a database or a taxonomy.

The entity linking problem is defined as follows: given a mention, $t \in \mathcal{T}$ and an ontology of unique entities, $\mathcal{Y}$, the objective is to learn a mapping function $f : t \mapsto \mathcal{Y}_i$ where $\mathcal{Y}_i$ is a unique entity in the ontology.

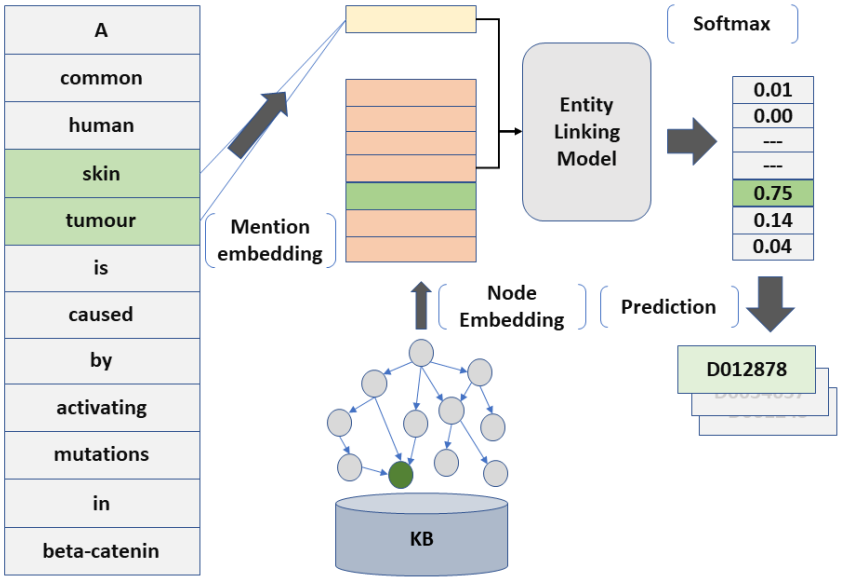


Figure 5.1: This figure shows the entity linking workflow. A mention embedding is generated from the text. Similarly, concept embeddings are generated using the taxonomy information. These embeddings are then forwarded to the entity linking model to give a probability distribution over all concept given the mention.

EL for diseases a.k.a. Disease Normalization is a challenging task due to the evolving conceptualization and nomenclature of diseases. Disease concepts are intrinsically composite, i.e can be divided into components based on the affected body part or origin, differentiating signs and symptoms, or the pathogens that cause the disease. The naming of the diseases are decided by the authors by a combination of these components.

The disease concept's intrinsic compositionality is exploited to form a structured disease KBs such as MeSH, UMLS, etc. The KBs used in our experiments are MeSH (section 3.2), which is a tree-structured hierarchical taxonomy, and OMIM. As we described in section 3.3, we limit our search only to the descriptors of the disease branch of MeSH tree and convert any other concepts to either a disease node in the tree or ignore it if such conversion is not possible.

The procedure we follow for disease normalization is shown in figure 5.1. We first identify the disease names in a given text and generate a single representation for both single or multi-word disease

names. We also generate MeSH taxonomy embedding using graph neural networks, namely *node2vec* and GCN. These embeddings are then processed in the EL model to .

In section 5.1, we describe the MeSH embedding generation methods, namely *node2vec* and GCN and their analysis. We describe our EL methods depending on the type of MeSH embedding in section 5.2 followed by section 5.3 on results and analysis, and section 5.4 for discussion.

5.1 Taxonomy embedding generation

The tree structure of the MeSH disease vocabulary is converted to an undirected graph by ignoring direction. We use the networkX python package [1] for building this graph.

We use two graph embedding techniques, namely *node2vec* (2.2.3) and GCNs (2.2.4) to generate representations of diseases that leverage the taxonomical structure of MeSH. *node2vec* is trained in an semi-supervised setting whereas GCN is learned from the upstream task.

5.1.1 node2vec encoding

The objective function of the *node2vec* framework is the same as that of a Skip-Gram model, and tries to maximize the log-probability of the neighboring nodes given the feature information of a node (eqn. 2.5). The *node2vec* framework is used to generate two types of representation of the diseases, namely *type*-I and *type*-II.

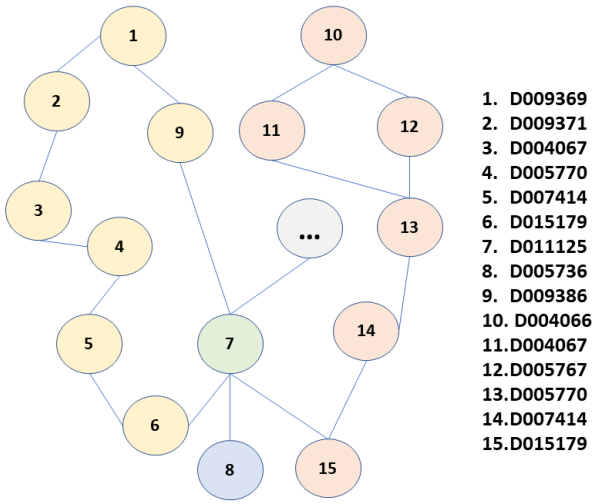


Figure 5.2: A portion of MeSH graph with our main focus on node 7 (D011125). For *node2vec*, any path that passes through node 7 will contribute towards *node2vec* node encoding, i.e. nodes 1, 9, 5, 6, 8, 14, 15, etc. might appear in the context window (of size 2). The same nodes i,e. 1, 9, 5, 6, 8, 14, 15, etc. will become the contributing factors in case of two layer GCN node encoding. Note that node 7 will have a self-loop in case of GCN encoding. The node without any number represents other neighboring nodes which are not included in this diagram.

Type-I *node2vec*: For the *type*-I formulation, we represent the disease nodes by their unique identifier. This implies that we do not include any lexical information about the disease as described in the *scope note* of the disease. Rather, we only use the taxonomy in the form of an undirected graph to learn a representation for the disease. For example, in figure 5.2, *node2vec* will generate random walks such as (4, 5, 6, 7, 8), (6, 7, 15, 14), (14, 15, 7, 8), (1, 9, 7, 6, 5), etc. The context window will have nodes such as 5, 6, 8, 15, 14, etc. which will contribute towards learning the node representation. Here, each of the nodes are initialized by a random vector and subsequently learned by the algorithm.

[1] https://networkx.github.io/

Type-II *node2vec*: In this case, we generate lexicalized embeddings for every disease node by learning a representation of the *scope note* using ELMo and taking the average over all the words. This embedding contains the lexical information from the description and the contextual information captured by using ELMo. These embeddings works as initial lexicalized representation of the nodes, i.e. initial value for the weight matrices in the Skip-Gram model.

We use a modified implementation of *node2vec* provided by the authors [2] and learn an embedding representation for the diseases. We implement our own Skip-Gram objective for training the taxonomy embedding over the random walks generated by *node2vec* using negative sampling, where we sample from the unigram distribution of the nodes.

Training details: In our *node2vec* embedding, we use the default settings as specified by the authors, except that we use random walks of length 10 for each source node and a total of 30 such random walks. We train both the *node2vec* type-I and type-II models for 20 epochs each in our implementation of Skip-Gram with SparseAdam and the ADAM optimizer respectively with initialized with a learning rate of 0.005, a batch size of 512, context window size of 2 and with negative samples of 5. By default, we set p and q values to 1. The *node2vec* based node encoding are generated as part of a preprocessing step before training EL models.

We validate the quality of the generated embedding by evaluating top 5 most similar embeddings to our query embedding. The most similar embeddings should also be neighbors in the MeSH graph. For *node2vec* type-I, the most similar MeSH ID of *D011125* are *D009386*, *D015179*, *D009377*, *D010580* and *D014402* which are neighbors in MeSH graph. Similarly, for *node2vec* type-II, most similar MeSH ID of *D011125* are *D015179*, *D044483*, *D003123*, *D006223* and *D004416* are also neighbors.

5.1.2 GCN encoding

In GCN encoding, we define our objective directly on the upstream task of predicting the correct concept among all the concepts in the KB given the mention. In contrast to *node2vec*, this is a supervised learning method. We proceed by generating an embedding for every disease node similar to *Type-II* node2vec. These embeddings, formulated in the form of the matrix are $\mathbf{X} = \mathbf{H}^{(0)}$ in equation 2.11. The parameters in the weight matrix in every layer of the GCN learns to combine information from neighboring nodes for each node in the MeSH taxonomy. Formally,

$$\mathbf{h}_v^{(j+1)} = \sigma\Big(\sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(j)} \mathbf{h}_u^{(j)} + b^{(j)}\Big) \tag{5.1}$$

where j indexes the layer, $\mathcal{N}(v)$ are neighbors of node v, $b^{(j)}$ is the bias and $\mathbf{h}_v^0 = \mathbf{x}_v$ is the embedding of the disease node v. The output of the last hidden layer and without the activation function is our new embedding representation of every disease node. We introduce randomness in the training following the authors (Kipf and Welling, 2016) by using dropout

For GCN encoding, we use a two-layered GCN, which implies that each node receives information from nodes at most two hops away. Note that our graph has no notion of weight, hence every node is equidistant from its immediate neighbors.

We validated our GCN embedding following the same procedure as *node2vec*. The top 5 most similar MeSH ID of *D011125* are *D044483*, *D005736*, *D018256*, *D010580* and *D003123*.

5.1.3 Analysis of embedding

In figures 5.3, 5.4 and 5.5, the encoded high-dimensional node embeddings are visualized *T-distributed Stochastic Neighbor Embedding* (T-SNE). It first converts the similarities between the data points to joint probabilities and then tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. We highlight the disease branch root and leaf nodes of MeSH tree in red and blue colors respectively.

In unlexicalized *node2vec* (fig. 5.3), we observe that general concept lies towards the center and as we move along any radii outwards the concepts become more specific. For instance, the disease node *Leukemia* and all is children follows this pattern. Notice that few children of disease *Leukemia* lies in a different region because they belongs to a more than one branch of disease root. Clearly, neighboring nodes in graphs are also closer in embedding space. The outer circle of embeddings mostly consist of leaf nodes. Lexicalized *node2vec* embeddings preserve the neighboring nodes and form small local

[2] https://github.com/aditya-grover/node2vec

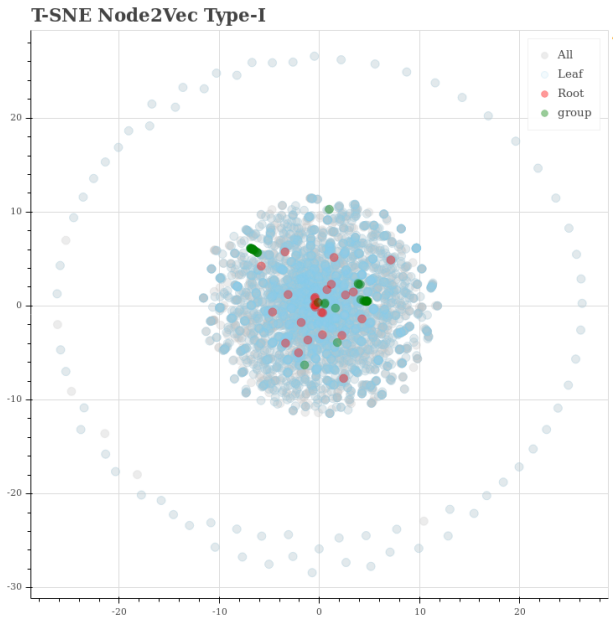


Figure 5.3: Visualization of *node2vec* Type-I MeSH embeddings. We used T-SNE plot highlighting the leaf nodes (skyblue) and root nodes (red) of disease MeSH Tree. We also plot the disease 'Leukemia' and all its children in green. The highly connected nodes or general concepts lies near the center and specific concepts in taxonomy have outward bound trajectory along radii centered at (0,0).

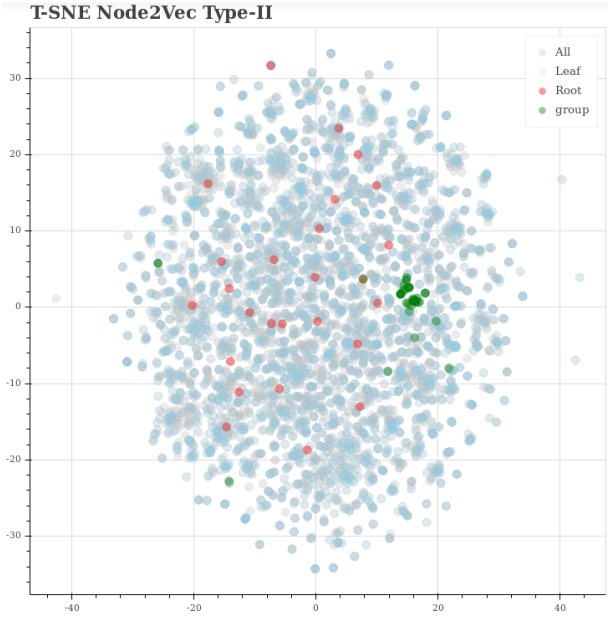


Figure 5.4: Visualization of *node2vec* Type-II MeSH embeddings. We used T-SNE plot highlighting the leaf nodes (skyblue) and root nodes (red) of disease MeSH Tree. We also plot the disease 'Leukemia' and all its children in green. Diseases with similar properties form clusters.

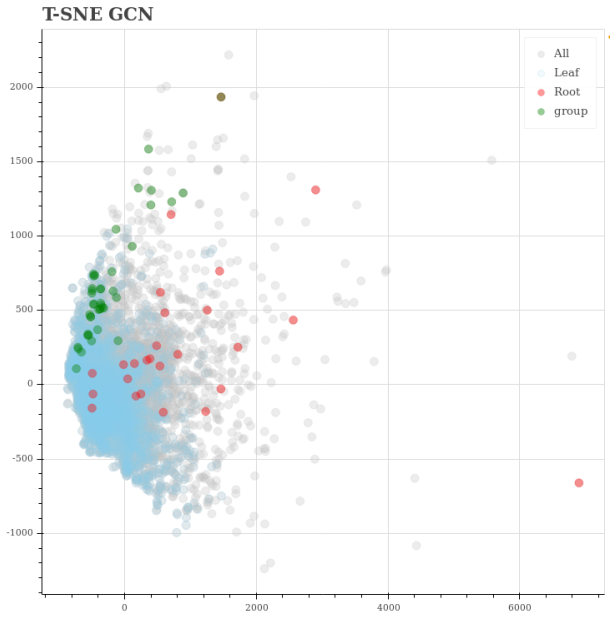


Figure 5.5: Visualization of GCN MeSH embeddings. We used T-SNE plot highlighting the leaf nodes (skyblue) and root nodes (red) of disease MeSH Tree. We also plot the disease 'Leukemia' and all its children in green. The leaf node diseases are located near (0,0) whereas root nodes and other internal nodes lies outwards.

regions of concepts In figure 5.4, the disease node *Leukemia* and its children forms a cluster. We did not observe any clear pattern like which concepts lie near the center (0,0) and outside. In the case of GCN embedding, we see concepts form a very very small group (2-3 nodes) in local regions. Also, we observe general concepts are usually spread outwards which is totally opposite of *node2vec* Type-I, i.e. the leaf nodes lies near (0,0) and other concept outwards.

5.2 Method

The model architecture used for EL is inspired by the fact that a entity mention in a text should be represented in the same embedding space as the canonical entity in the target KB, and should be thus most similar (near) to the correct target entity than all other entities in the KB. Our EL models are based on the above assumption, i.e. locality of embeddings. In addition, the context in which the entity appears should have a high correlation to the description of the entity in the KB.

Mentions can be represented as the embedding of a single or multi-word entity. These embeddings could be the output of the BiLSTM layers of the NER model (section 4.1). However, the NER model is trained to predict tags which is a classification task, and hence it may lose all the lexical and semantic information. This will not work for entity linking as we need this information intact to have a meaningful representation of the mention. We thus use ELMo to obtain the embedding for a single or multi-word span of a mention and use the average as our representation of the mention. The embedding of entities in the KB is generated by the taxonomy encoding schemes described in section 5.1.

Our EL model (figure 5.6) using the *node2vec* encoding scheme is a linear transformation layer with an objective function that maximizes the log-probability of a target concept given the features of a mention. In other words, we try to maximize the similarity between the mention embedding and target entity graph embedding (and conversely minimize similarity to any other entity in the KB). The linear transformation is necessary to adhere to our assumption that the local region of the mentions and entities should be the same in the embedding space.

Let $\mathcal{Y} = (y_1, y_2, \ldots, y_l)$ be all the entities present in the MeSH KB. Given a mention embedding $\mathbf{x} \in \mathbb{R}^m$ for a mention x and a entity embedding $\mathbf{y}_i \in \mathbb{R}^e$ and its corresponding concept $y_i \in \mathcal{Y}$, we do a linear transformation of the mention embedding $\mathbf{x}$ parameterized by $\mathbf{W} \in \mathbb{R}^{m \times e}$ and maximize the

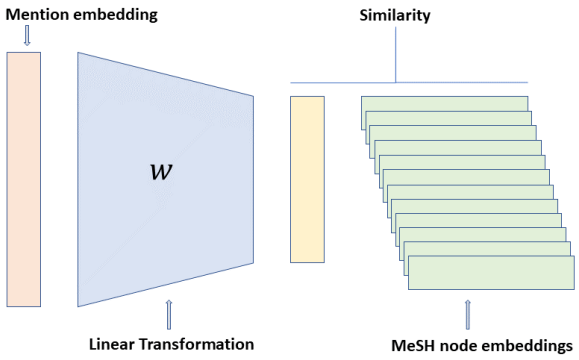


Figure 5.6: EL model using *node2vec* MeSH encoding is a linear transformation of mention embeddings followed by a similarity measure (dot product) with every other entity in the KB.

following objective function

$$\log p(\mathcal{Y} = y_i|\mathbf{x}) = \log\left(\frac{\exp\left(\mathbf{x^T W y}_i\right)}{\sum_{j=1}^{l}\exp\left(\mathbf{x^T W y_j}\right)}\right). \tag{5.2}$$

.

For our GCN node encoding, the node embeddings are learned on the EL task. Considering the fact that less number of parameters helps is a generalization, we don't apply any linear transformation to mention embedding rather we rely on the GCN to maintain our embedding space assumption. In other words, we directly use GCN as our EL model which is trained to learn MeSH node representation in favour of the EL task. Since our mention embeddings are average of ELMo embeddings of span of mention words, we assume that these embeddings, i.e disease names embeddings, will not differ much in its various occurrence in different sentences.

Given a mention embedding $\mathbf{x}$ and corresponding entity $y_i \in \mathcal{Y}$, we encode the entity using GCN g parameterized by θ, i.e. $\mathbf{y}_i = g(y_i; \theta)$. Our objective function maximizes the log-probability of the target concept given the features of the source mention. Formally,

$$\log p(\mathcal{Y} = y_i|\mathbf{x}) = \log\left(\frac{\exp\left(\mathbf{x^T} g(y_i;\theta)\right)}{\sum_{j=1}^{l}\exp\left(\mathbf{x^T} g(y_j;\theta)\right)}\right) \tag{5.3}$$

5.3 Result and Analysis

Evaluation metrics: The standard metric used in most of the literature for disease normalization is *precision*, *recall* and *F1-score*. This metric is meaningful if we have a pipeline architecture or we are using a NER system first to identify the named entity and normalizing them to concepts. In doing so we might however identify false positives and miss out false negatives, errors that would propagate to the global precision, recall, and F1-scores of EL system. In our case, we want to test the quality of our EL system independently of the NER system. This makes the accuracy score evaluated directly with respect to gold set a justifiable metric in this context. We also report micro F1-score to be consistent with reporting the results of entity linking task (Ganea and Hofmann, 2017). Accuracy score measures the total number of correctly mapped mentions divided by a total number of mentions in the text. Also, if we rank all our entities in the KB in terms of their similarity score with respect to mention and/or prediction confidence, we get accuracy @k or precision @k by considering an entity to be correctly identified if the correct entity appears within k^{th} rank. We also evaluate mean reciprocal rank (MRR) defined as

$$MRR = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{rank_i} \tag{5.4}$$

where m is total number of mentions in the test set and $rank_i$ is the rank at which the correctly mapped concept appears in the sorted list of all entities based on similarity score/prediction confidence for the i^{th} mention.

Parameters	Default Values
node2vec	
Node embedding size	1024
p	1
q	1
number of linear layers	1
GCN	
number of GCN layers	2
hidden dimension	2048

Table 5.1: Default values of parameter for all the experiment unless specified.

Model	Parameter	MRR ↑	F1-score ↑	Acc @1 ↑	Acc @5 ↑	Acc @30 ↑
***node2vec* Unlexicalized**						
Type-I*	default	0.749 ± 0.002	0.718 ± 0.004	0.72 ± 0.004	0.778 ± 0.002	0.819 ± 0.006
Type-I	p = 2	0.753 ± 0.002	0.723 ± 0.004	0.725 ± 0.003	0.782 ± 0.003	0.817 ± 0.003
Type-I	q = 2	0.744 ± 0.003	0.722 ± 0.007	0.722 ± 0.005	0.776 ± 0.004	0.812 ± 0.003
***node2vec* Lexicalized**						
Type II	default	0.757 ± 0.001	0.721 ± 0.004	0.724 ± 0.001	0.791 ± 0.004	0.842 ± 0.004
GCN						
GCN	default	0.744 ± 0.006	0.71 ± 0.008	0.71 ± 0.007	0.781 ± 0.009	0.831 ± 0.005

Table 5.2: Results on test set of our EL models for different type of MeSH encoding. The setting used are default parameters mentioned in table 5.1 unless explicitly mentioned. Each experiment is run of 5 times and we report the mean and the standard deviation of the results. * We consider unlexicalized *node2vec* as our baseline. Acc denotes accuracy

We select our models using early stopping based on the accuracy@1 after each epoch on the validation set for each of the run in an experiment. We then report the mean and standard deviation of our results on the test set by evaluating on each run in an experiment.

Training details: We train all our EL neural networks using back-propagation and SGD with ADAM optimizer, the learning rate of 0.001 and the batch size is 32. Our EL models using *node2vec* MeSH encoding are trained for 100 epochs whereas GCN based EL model for 500 epochs. The node embedding size we use for all our models is 1024. The EL model with *node2vec* encoding is a linear transformation with the same input and output dimension of 1024. This allows the model to learn the identity mapping if required. The EL model based on GCN encoding comprises the two-layer GCN with the first layer output dimension and second layer input dimension equal to 2048, and second layer output dimension to 1024.

5.3.1 Quantitative Analysis

In table 5.2, we report the result of evaluation on the test set. Our baseline model which relies on an unlexicalized *node2vec* encoding of the MeSH taxonomy has a mean MRR score of 0.749, F1-score of 0.718, while accuracy @1, accuracy @5 and accuracy @30 is 0.72, 0.778 and 0.819 respectively. Increasing the p-value to induce the random walk not to revisit the node again in *node2vec* increases MRR slightly by 0.004 and a similar increase is seen with F1-score, accuracy @1 and accuracy @5. Also, if we bias the walks to visit nodes closer results in a slight drop in performance.[3] Using lexicalized *node2vec* gives rise to the best performing model with mean MRR of 0.757, accuracy@5 of 0.791 and accuracy @30 of 0.842. GCN encoding is similar to our baseline expect by its accuracy @30. Comparing *node2vec* type-II to other models we can say that although the accuracy @1 is nearly equal, the method is able to rank the correct concepts higher on average in comparison to other models as evident from accuracy @15 and accuracy @30, besides MRR. This explanation is valid for GCN encoding as well. The commonality between these two models is the lexicalized node embedding using ELMo which could be the key contributing factor for the improvement rather than the differences in the model architectures.

[3]These performance increases or drops are not significant enough to claim that one approach is better than the other. Many more runs of the experiment would be required and a better, non-random parameter search procedure.

Model	MRR ↑
node2vec Type-I	0.153 ± 0.006
node2vec Type-II	0.172 ± 0.005
GCN	0.176 ± 0.007

Table 5.3: We evaluated the MRR of entities that did not appeared in the training set for all the models. We run each model 5 times on the test set and report the mean and standard deviation.

We also evaluate how these models perform in zero-shot learning scenario 5.3. We selected only those entities that are unique to test set and evaluated the MRR for our models. Our baseline model gives a MRR of 0.153, whereas Lexicalized *node2vec* based EL model and GCN based model gives 0.172 and 0.176 and hence performs better.

5.3.2 Qualitative Analysis

We analyze few results of the mapping functions by randomly picking few examples of our baseline models and compared with other models. In case of the entity *D007634*, the baseline EL model predicts *D030342* followed by *D000853* and the correct entity is ranked at 1231. *D007634* is a eye disease and predicted *D030342* node has a sibling disease related to eye, and *D000853* is a grand-child of eye disease branch. Lexicalised *node2vec* embedding based EL model also predicts *D030342* followed by a different entity *D012174* which is also a grand-child of eye disease, and rank the correct entity at a higher rank of 1582. In case of GCN based model, the predictions are very different, i.e it predicts *D000567* and *D003744* both of which are unrelated to eye disease, and ranking the correct entity at 3889 position.

In another example, though the predicted tag should be *D015458* (Leukemia, T-Cell), the baseline EL model predicts *D016399* (Lymphoma, T-Cell), and *D017728* (Lymphoma, Large-Cell, Anaplastic), and lexicalised *node2vec* based EL model predicts *D016399* (Lymphoma, T-Cell) and *D007938* (Leukemia). The lexicalised *node2vec* model ranks the correct tag higher than baseline model. The GCN based model predicts *D007945* (Leukemia, Lymphoid) and correct;y predicts at rank 2.

5.4 Discussion

From our analysis on different graph embeddings, we observe different distribution on graph nodes and different scale in the T-SNE plots (figure 5.3, 5.4, 5.5). Though T-SNE plots does not depict the exact behaviour of nodes in the actual embedding space, we can however make approximation considering the fact that every model follows the same set of procedures. Since lexicalized *node2vec* based embedding are sparse with nodes forming clusters of concepts, we can say they perform better than other models. The sparsity of GCN is not consistent which could be the reason of lower accuracy @1 but higher accuracy @30. Another factor could be the use of ELMo embedding in lexicalized *node2vec* and GCN.

Instead of directly formulating our approach using the graph embeddings for disease normalization, we could use this procedure for candidate generation and/or use any other ranking algorithm from information retrieval literature. In previous approaches, many procedures used coherence models for predicted entities. This allows the models favor concepts that are likely to occur in similar context within an abstracts rather than concepts that are unrelated. Incorporating such models into our architecture would also enhance the performance of the model.

The size of the dataset is very small as compared to the dataset used in other NLP task using deep learning. Currently, our EL model is a simple linear layer along with ELMO and graph embeddings. We can use other sophisticated architecture provided we have more annotated data.

Chapter 6

Multi-task Learning

Multi-task learning can be viewed as a form of inductive transfer that can help a model by introducing an inductive bias, which causes a model to have more preference towards some hypothesis over others. In a way, it reduces the risk of overfitting as well as models the ability to fit random noise. In the context of deep learning, there are typically two ways of performing multi-task learning. In one case, the hidden layers of the neural networks are shared between all the tasks, while keeping the task-specific output layer. This approach is known as *hard parameter sharing*. On the other hand, each task has its own model with its own parameters where the distance between the parameters of the model is regularized to encourage the parameters to be similar. This is known as *soft parameter sharing*.

Hard parameter sharing is still the standard two decades after its inception in 1993 by Caruana (1997). The sharing works if the tasks are closely related, or otherwise, there is no guarantee (Ruder, 2017) that performance will improve. There are many choices of related or auxiliary tasks depending on the problem. In the context of NER, part-of-speech tagging can be for example used as an auxiliary task.

The problem of disease normalization is tackled by first identifying the disease names (NER) and then normalizing them (EL) as depicted in figure 6.1. The implicit relation we have here is that only the span of mention identified by NER is used for EL. We can use this as a signal for training both tasks simultaneously.

6.1 Method

We use hard parameter sharing in our model architecture for multi-tasking where we share a layer of BiLSTM among the task specific layers to represent features useful for both the individual model architectures. The basic model architecture is as follows:

Given a sentence, $\mathbf{x}_{1:n} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ where each $\mathbf{x}_i$ is a word embedding, a BiLSTM is used to encode better representations of the words driven by the upstream tasks. The output of the BiLSTM is then fed to task-specific models of NER and EL respectively. The NER model and EL model are the same (or similar) to those described in previous chapters.

$$\hat{\mathbf{x}}_{1:n} = \text{encode}(\mathbf{x}_{1:n}) = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_n) \tag{6.1}$$

$$\text{NER}(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_n) = (y_1, y_2, \ldots, y_n) = \mathbf{y}_{1:n} \tag{6.2}$$

where y_i are NER tags at i^{th} position.

$$\text{EL}(\hat{\mathbf{x}}_{i\ldots j}) = t_k \tag{6.3}$$

where $\hat{\mathbf{x}}_{i\ldots j}$ represents the average of the encoded vectors from position i^{th} to j^{th} of the span of mention and t_k is the predicted entity from the entities $\mathbf{T}$ in KB.

The objective functions of the NER and EL tasks are the same as eqn. 4.4 and eqn. 5.3 respectively.

$$L_1 = -s(\hat{\mathbf{x}}_{1:n}, \mathbf{y}_{1:n}) + \log\left(\sum_{\hat{y}_{1:n} \in \mathbf{Y}_{(\mathbf{x})}} \exp\left(s(\mathbf{x}_{1:n}, \hat{y}_{1:n})\right)\right) \quad \text{and} \quad L_2 = -\sum\left(\mathbf{x}\hat{\mathbf{t}}_k - \log\sum_{j=1}^{l}\exp\left(\mathbf{x}\hat{\mathbf{t}}_j\right)\right)$$
$$loss = L_1 + L_2 \tag{6.4}$$

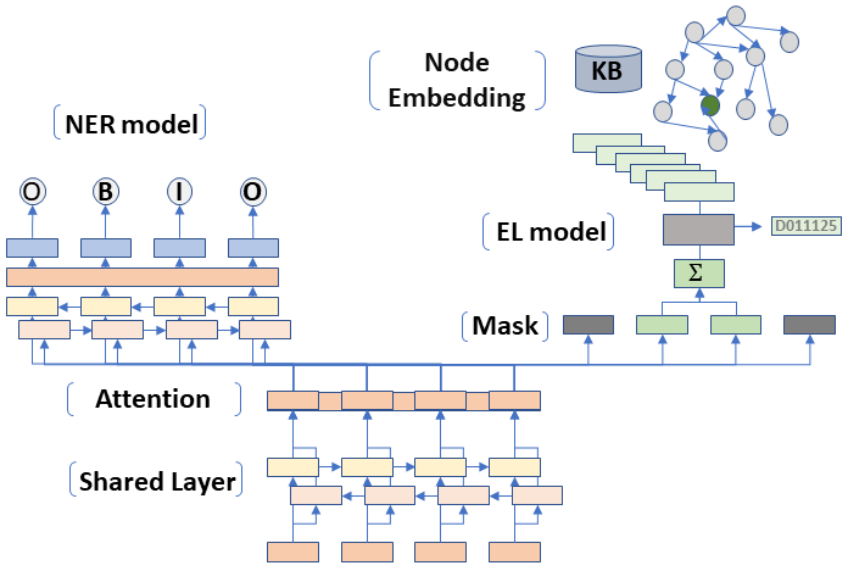


Figure 6.1: Visualization of the Multi-task learning architecture. A layer made of BiLSTM is shared between the two task, namely NER and EL. The output of the shared layer is directly sent to NER model, whereas only the disease names position are sent to EL model by using a mask. Attention layer is placed on top of the shared layer to study the behaviour and contributions of different activation functions for MTL.

where $\mathbf{Y}_{(\mathbf{x})}$ represents all possible tag sequences for a sentence $\mathbf{X}$, s is a similarity function, $\hat{\mathbf{t}}$ is encoded representation of entity and L_2 is the sum of loss over all mention in a sentence.

This basic model architecture is modified by implementing a few changes as follows:

Attention: Using the attention mechanism in a neural network allows the model to focus on essential parts of the network based on some additional information. We introduce an attention layer after the shared layer to re-weight the encoded representation of the word. We use a linear layer to evaluate a score for each word followed by applying an activation function to obtain the weights of each word representations. These weights are then multiplied with the encoded representation to obtain the re-weighted encoding.

$$\begin{gathered}\mathbf{S} = \text{score}(\hat{\mathbf{x}}_{1:n}) = \text{linear}(\hat{\mathbf{x}}_{1:n}; \mathbf{W}) \quad \text{and} \quad \text{weights} = \text{activation}(\mathbf{S}) \\ \text{re-weighted } \hat{\mathbf{x}}_{1:n} = \text{weights} \times \hat{\mathbf{x}}_{1:n}\end{gathered} \tag{6.5}$$

where $\hat{\mathbf{x}}_{1:n} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ is a sentence and $\mathbf{x}_i \in \mathbb{R}^d$ is a word embedding, and $\mathbf{W} \in \mathbb{R}^{d \times 1}$ is a weight matrix.

We experimented with four kinds of activation function, namely *softmax*, *sparsemax*, *fusedmax* and *oscarmax* (Niculae and Blondel, 2017). The most commonly used activation function used in attention mechanism is softmax which gives dense attention weights, i.e. all elements receive a small weight no matter if are relevant for decision making. This limitation is overcome by using sparsemax that put zero weights to a certain position which can never be achieved by using softmax. Fusedmax activation encourages the network to pay attention to continuous segments in a sequence whereas oscarmax encourages the network to pay equal attention to possibly non-contiguous groups of words.

6.2 Result and analysis

Evaluation: We used the same metric for NER and EL defined in 4.2 and 5.3 respectively. That is *precision*, *recall* and *F1-score* for NER, and *Mean Reciprocal Rank* and *accuracy*.

We select our models using early stopping based on the accuracy@1 for EL model in MTL setting an experiment. In case of training on individual task, i.e. NER and EL, we use F1-score and accuracy@1 respectively. We report our results on test set.

Parameters	Default Values
Sentence split	✓
Shared and NER layers	
Word embedding (ELMO)	1024
Word RNN units	512
Dropout NER	0.5
Dropout EL	0.2
EL GCN	
number of GCN layers	2
hidden dimension	2048

Table 6.1: Default values of parameter for all the experiment unless specified.

Model	Precision ↑	Recall ↑	F1-score ↑	MRR ↑	Accuracy @1 ↑	Accuracy @30 ↑
No Attention						
NER & EL	0.877	0.884	0.881	0.749	0.717	0.817
Only NER	0.884	0.87	0.877	-	-	-
Only EL	-	-	-	0.746	0.717	0.816
Attention						
Softmax	0.882	0.823	0.851	0.745	0.714	0.813
Fusedmax	0.873	0.861	0.867	0.739	0.703	0.825
Oscarmax	0.87	0.813	0.841	0.734	0.708	0.815
Sparsemax	0.883	0.726	0.797	0.74	0.704	0.836

Table 6.2: Results on test set of our MTL experiment with or without attention layer after shared layer. We report precision, recall and F1-score for NER task, and for EL we report MRR and accuracy. We run each experiment only once and report out results for different activation functions. We also perform an ablation study of each task by training without the attention layer.

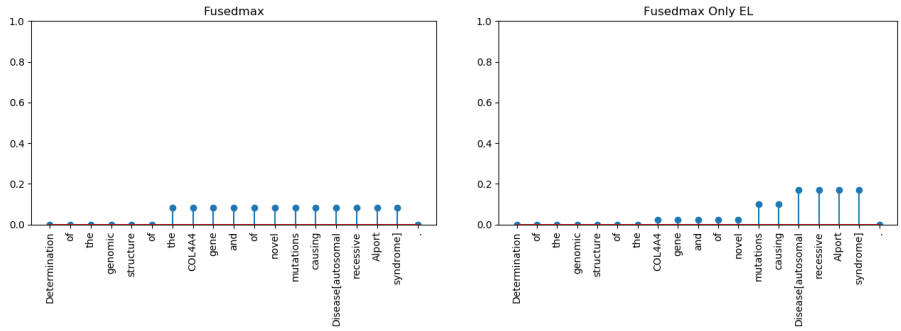
Training details: We again use SGD with ADAM optimizer with learning rate 0.001, batch size 32 and run for 500 epochs. We use ELMo embedding as our initial word representation. All our BiLSTMs layers in the network architecture, namely shared layer and NER layer, have 512 hidden units resulting in a 1024 dimensional output vector which is same as the size of ELMo embedding size. Variational dropout is used for the shared and NER layer with dropout probability 0.2 and 0.5 respectively. We use self-attention for our NER layer and experiment with different activations for attention after the shared layer. We evaluate scores for attention using a linear layer of input size 1024 and 1-dimensional output. We use this github[1] implementation of different activation functions. Our EL model is based on the GCN encoding as described in 5.3. Each of our experiment is run only once and we report our results on test set in the table 6.2[2]. We fix the same random seed for all our experiment to have the same initial starting condition.

6.2.1 Quantitative Analysis

We experiment with different settings of attention, namely without or with attention layer, and the four types of activation function. In our setting without the attention layer, we also perform each task independently to know our model performance on an individual task. The best overall performing model is without the attention layer in which NER has an F1-score of about 0.881 and MRR of 0.749. As compared to training the model individually on each task, multi-tasking settings show only a slight improvement. In our experiment with different activation functions for attention, fusedmax performs better in NER whereas softmax performs better for EL. If we consider looking for the correct candidates among the top 30 predictions for EL, then sparsemax performs the best with an accuracy score of 0.836 followed by fusedmax.

[1] https://github.com/vene/sparse-structured-attention

[2] The very long duration of each experiment and the lack of very large computational resources – multitasking is extremely resource-greedy – prevented us from running it multiple times.



(a) Fusedmax (b) Fusedmax only EL

Figure 6.2: We plot the attention weights using fusedmax activation function for MTL tasks, trained and tested on both NER and EL task in figure 6.2(a). The disease name as well as few nearby words is allocated higher weights, and rest zero. In the figure 6.2(b), we plot the attention weights when the MTL model is only trained and tested only on EL task. The disease name is allocated higher weights followed by few nearby words, and rest of the words are allocated zero weights.

6.2.2 Qualitative Analysis

We analyze the attention weights used in different activation functions, namely fusedmax, oscarmax and sparsemax. In figure 6.2(a), the attention weighs near to the disease name have a large weight and rest are zeroed out. We observe that the only contributing factor for this behavior is the EL task. Figure 6.2(b) shows a similar plot of training only our EL part of the model with fusedmax where disease words got the highest attention weights followed by nearby words, and so on up until only zeros are left at the beginning of the sentence. A similar study where we trained only for NER did not show any such pattern. In fact, every word except the end of the sentence is given equal attention weights. Oscarmax does not show any pattern and give equal weights to every word in the sentence. Sparsemax shows a similar behavior like oscarmax with equal weight to every word. However, when we train only for EL it allocates a large weight on one of the disease words as shown in figure 8.1.

6.3 Discussion

We experimented with MTL to leverage the common signal between both the task, i.e. the identification of the boundary of the disease names and extracting features favorable for normalization. However, our training pipeline is very restrictive resulting in very long training time. As a result, we could only perform one experiment for each of the different settings. We tried to reduce the variation in training procedure by fixing the random seeds.

Our MTL models performs better as compared to training the models only on individual task. This could be because we use a large model while on a single task it overfits on training set. Whereas in case of MTL, signals from both the task reduces this overfitting.

In order to improve the model, we need to improve the training pipeline. We should also search for good set of parameters and perform multiple experiments for each experiment to do a fair comparison between models.

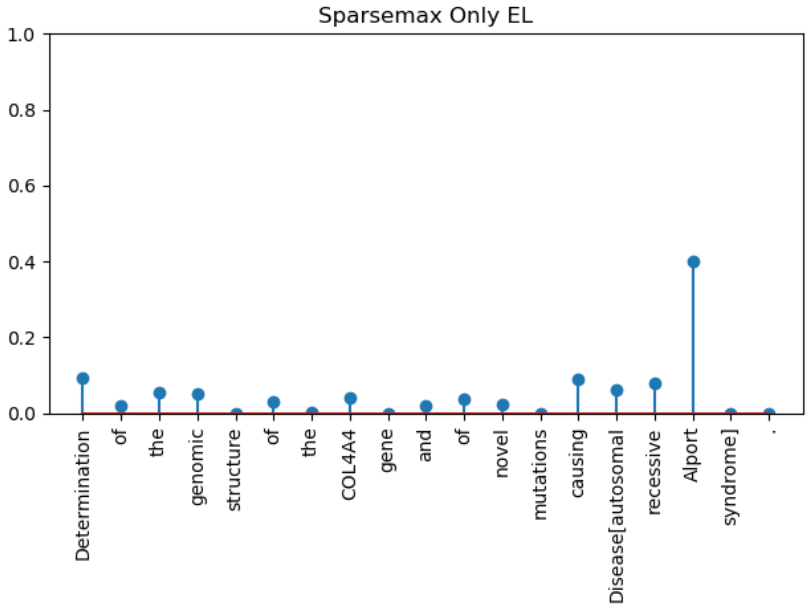


Figure 6.3: We plot the attention weights using sparsemax activation function trained and tested only on the EL task in the MTL architecture. We see that the word *Alport* of the disease name *autosomal recessive Alport syndrome* is alloted maximum weights as compares to other words in the disease name as well as rest of the text.

Chapter 7

Conclusion and Future work

In this thesis, we tried to address the problem of disease normalization by adapting state-of-the-art open-domain neural network architectures in the disease domain. Usually, a disease normalization system would first identify the disease name using *Named Entity Recognition* followed by linking the identified entity to a controlled database, i.e *Entity Linking*. We implemented a NER that allows identifying disease names in a document with a performance on par with the state-of-the-art model in disease domain. Specifically, we adapted the basic architecture proposed by Lample et al. (2016) and Ma and Hovy (2016) and used ELMo embeddings for representing the words in a text document. In addition, we experimented with self-attention, dynamic re-weighing of word-level and character-level embedding, and recurrent dropout to enhance the performance and model interpretability. We observed that ELMo embeddings lead to substantial improvement in NER performance as compared to other look-up table based embeddings.

We approach the EL problem as a mapping function that maps entities identified in the text to the concepts in the controlled vocabulary (KB). In contrast to previous approaches, we use graph neural networks for encoding the concepts in the KB that uses both the structural and semantic property of diseases. The entities in the text and the disease description are represented using ELMo embeddings that capture the contextual semantics of disease names assisting the model in disambiguation. Specifically, we use unlexicalized *node2vec* encoding the disease nodes in the MeSH taxonomy, represented as an undirected graph, as our baseline encoding model. We incorporate the lexical information from the disease description provided in the MeSH, by initializing the node with these as an embedding, giving a new lexicalized *node2vec* encoding model. In addition, we use graph convolution networks using the same lexicalized node representation giving our 3rd encoding model. The *node2vec* based node encoding model generates node representation in a preprocessing step followed by the EL model which is a linear transformation. The GCN based encoding model is directly trained on the EL task, i.e. we do not need a separate model of EL. Our lexicalized *node2vec* based EL model achieves better performance as compared to the baseline model and GCN. The lexicalized *node2vec* and GCN rank the correct entities higher as compared to our baseline. Thus we can say node lexicalization does improves over unlexicalized counterpart.

We also experimented with Multi-task learning to leverage the common signal between both the NER and EL task. The NER task tries to identify the boundary of the span of disease names whereas the EL task requires the disease name to have a representation that assists in disambiguation. We use a shared layer common to both the task and task-specific layers for NER and EL. We achieve better results in MTL setting as compared to training in the models on individual tasks. We also experiment with attention mechanism with different activation function, namely *softmax*, *fusedmax*, *oscarmax* and *sparsmax* to assist the boundary identification task more appropriately. However, results obtained did not improve over the basic model without the attention layer.

Using the NER and EL model in a pipeline architecture, our disease normalization does not give a satisfactory model as compared to previous approaches. The reasons could be the lack of large training dataset that is required for deep learning models to achieve good performance, which otherwise results in overfitting. In addition, we did not explore the hyper-parameter space enough to identify the most favorable set of parameters for our task. We believe having a large dataset with efficient parameter search methods, we could achieve better results for disease normalization.

Our graph-based disease node encoding is the first of its kind, to best of our knowledge. The disease node in the MeSH taxonomy has many attributes of which we only incorporated the disease description

in our initial node representation. One of the many attributes is disease synonyms that are the basis for many dictionary and rule-based disease normalization techniques. In the future, we could explore using these synonyms of the diseases in our node encoding architecture as well. Another approach is to represent the description of the disease node more robustly, for example by taking inspiration from denoising auto-encoder, instead of taking an average of ELMo word embeddings.

Our MTL also shows promising results as compared to results achieved individually on each task. However, we experimented each model only once because of inefficient training pipeline. We are also constrained by small dataset and lack of hyper-parameter search which could be taken up in future. MTL has potential but we haven't found the ideal setup yet. In addition, the idea behind the use of different activation function for assisting the boundary-making decision could be interesting. Sparsity, though easy to motivate and better interpretable did not work well. But more work can be done in future. Perhaps changing the composition function before and/or after the sparse attention layers could be an option.

Bibliography

Ashraf, A., Khan, S. S., Bhagwat, N., Chakravarty, M. M., and Taati, B. (2018). Learning to unlearn: Building immunity to dataset bias in medical imaging studies. *CoRR*, abs/1812.01716.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2008). Dbpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pages 722–735.

Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Chapelle, O., and Weinberger, K. (2010). Learning to rank with (a lot of) word features. *Inf. Retr.*, 13(3):291–314.

Beltagy, I., Cohan, A., and Lo, K. (2019). Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676.

Bikel, D. M., Schwartz, R. M., and Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Borthwick, A. E. (1999). *A Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, New York, NY, USA. AAI9945252.

C. Bunescu, R. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation.

Carreras, X., Màrquez, L., and Padró, L. (2002). Named entity extraction using AdaBoost. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Caruana, R. (1997). Multitask learning. *Mach. Learn.*, 28(1):41–75.

Cho, H., Choi, W., and Lee, H. (2017). A method for named entity normalization in biomedical articles: application to diseases and plants. *BMC Bioinformatics*, 18(1):451.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398.

Curran, J. R. and Clark, S. (2003). Language independent NER using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 164–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Davis, A. P., Wiegers, T. C., Rosenstein, M. C., and Mattingly, C. J. (2012). MEDIC: a practical disease vocabulary used at the Comparative Toxicogenomics Database. *Database*, 2012.

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

dos Santos, C. and Guimarães, V. (2015). Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33, Beijing, China. Association for Computational Linguistics.

Doğan, R. I., Leaman, R., and Lu, Z. (2014). NCBI disease corpus. *J. of Biomedical Informatics*, 47(C):1–10.

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding back-translation at scale. *CoRR*, abs/1808.09381.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134.

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

Ganea, O. and Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. *CoRR*, abs/1704.04920.

Gers, F. A. and Schmidhuber, J. (2000). Recurrent nets that time and count. Technical report.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, International Convention Centre, Sydney, Australia. PMLR.

Grishman, R. and Sundheim, B. (1996). Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA. Association for Computational Linguistics.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653.

Habibi, M., Weber, L., Neves, M., Wiegandt, D. L., and Leser, U. (2017). Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48.

Han, X., Sun, L., and Zhao, J. (2011). Collective entity linking in web text: A graph-based method. In *in: Proceedings of the 34th international Conference on Research and Development in Information Retrieval*, pages 765–774.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., and Wang, H. (2013). Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–34, Sofia, Bulgaria. Association for Computational Linguistics.

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

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, Stroudsburg, PA, USA. Association for Computational Linguistics.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Jimeno, A., Jimenez-Ruiz, E., Lee, V., Gaudan, S., Berlanga, R., and Rebholz-Schuhmann, D. (2008). Assessment of disease named entity recognition on a corpus of annotated sentences. *BMC Bioinformatics*, 9(3):S3.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Kulkarni, S., Singh, A., Ramakrishnan, G., and Chakrabarti, S. (2009). Collective annotation of Wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 457–466, New York, NY, USA. ACM.

Kuru, O., Can, O. A., and Yuret, D. (2016). CharNER: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921, Osaka, Japan. The COLING 2016 Organizing Committee.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Leaman, R. (2008). BANNER: an executable survey of advances in biomedical named entity recognition. In *Pacific Symposium on Biocomputing*, pages 652–663.

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

Leaman, R. and Lu, Z. (2016). TaggerOne: joint named entity recognition and normalization with semi-Markov Models. *Bioinformatics*, 32(18):2839–2846.

Li, H., Chen, Q., Tang, B., Wang, X., Xu, H., Wang, B., and Huang, D. (2017). CNN-based ranking for biomedical entity normalization. *BMC Bioinformatics*, 18.

Lin, Y.-F., Tsai, T.-H., Chou, W.-C., Wu, K.-P., Sung, T.-Y., and Hsu, W.-L. (2004). A maximum entropy approach to biomedical named entity recognition. In *Proceedings of the 4th International Conference on Data Mining in Bioinformatics*, BIOKDD'04, pages 56–61, Berlin, Heidelberg. Springer-Verlag.

Lou, Y., Zhang, Y., Qian, T., Li, F., Xiong, S., and Ji, D. (2017). A transition-based joint model for disease named entity recognition and normalization. *Bioinformatics*, 33(15):2363–2371.

Lowe, D. M. and Sayle, R. A. (2015). LeadMine: a grammar and dictionary driven approach to entity recognition. *Journal of Cheminformatics*, 7(1):S5.

Luo, G., Huang, X., Lin, C.-Y., and Nie, Z. (2015). Joint named entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Ma, X. and Hovy, E. H. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *CoRR*, abs/1603.01354.

McCallum, A. and Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA. Association for Computational Linguistics.

McNamee, P. and Mayfield, J. (2002). Entity extraction without language-specific resources. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Munro, R. and Manning, C. D. (2012). Accurate unsupervised joint named-entity extraction from unaligned parallel text. In *Proceedings of the 4th Named Entity Workshop*, NEWS '12, pages 21–29, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26. Publisher: John Benjamins Publishing Company.

Nguyen, D. B., Theobald, M., and Weikum, G. (2016). J-NERD: Joint named entity recognition and disambiguation with rich linguistic features. *Transactions of the Association for Computational Linguistics*, 4:215–229.

Niculae, V. and Blondel, M. (2017). A regularized framework for sparse and structured neural attention. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3338–3348. Curran Associates, Inc.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

Ramshaw, L. A. and Marcus, M. P. (1995). Text chunking using transformation-based learning. *CoRR*, cmp-lg/9505040.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rei, M., Crichton, G., and Pyysalo, S. (2016). Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318, Osaka, Japan. The COLING 2016 Organizing Committee.

Reimers, N. and Gurevych, I. (2017). Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

Sahu, S. and Anand, A. (2016). Recurrent neural network models for disease name recognition using domain invariant features. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2216–2225, Berlin, Germany. Association for Computational Linguistics.

Shen, W., Wang, J., and Han, J. (2015). Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460.

Sil, A. and Yates, A. (2013). Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2369–2374, New York, NY, USA. ACM.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. ACM.

Sun, C., Guan, Y., Wang, X., and Lin, L. (2007). Rich features based conditional random fields for biological named entities recognition. *Computers in biology and medicine*, 37 9:1327–33.

Sun, Y., Lin, L., Tang, D., Yang, N., Ji, Z., and Wang, X. (2015). Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1333–1339. AAAI Press.

Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4(4):267–373.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Daelemans, W. and Osborne, M., editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wei, Q. and Dunbrack, Jr, R. L. (2013). The role of balanced training and testing data sets for binary classifiers in bioinformatics. *PLOS ONE*, 8(7):1–12.

Wright, D., Katsis, Y., Mehta, R., and Hsu, C.-N. (2019). Normco: Deep disease normalization for biomedical knowledge base construction. In *Automated Knowledge Base Construction*.

Yamada, I., Shindo, H., Takeda, H., and Takefuji, Y. (2016). Joint learning of the embedding of words and entities for named entity disambiguation. *CoRR*, abs/1601.01343.

Zhang, W., Sim, Y. C., Su, J., and Tan, C. L. (2011). Entity linking with effective acronym expansion, instance selection and topic modeling. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 1909–1914. AAAI Press.

Zhao, S., Liu, T., Zhao, S., and Wang, F. (2018). A neural multi-task learning framework to jointly model medical named entity recognition and normalization. *CoRR*, abs/1812.06081.

Zhao, Z., Yang, Z., Luo, L., Wang, L., Zhang, Y., Lin, H., and Wang, J. (2017). Disease named entity recognition from biomedical literature using a novel convolutional neural network. *BMC Medical Genomics*, 10.

Zheng, Z., Li, F., Huang, M., and Zhu, X. (2010). Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 483–491, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chapter 8

Appendix

Model	Precision	Recall	F1-score	Accuracy@1
Leaman et al. (2013)	0.80	0.763	0.782	0.872
Wright et al. (2019)	0.863	0.818	0.840	0.878

Table 8.1: Disease Normalization scores of known systems on NCBI dataset

Model	Precision	Recall	F1-score
Habibi et al. (2017)	0.8643	0.8292	0..8464
Beltagy et al. (2019)	-	-	0.8691
BioFLARE[1]	-	-	0.8885

Table 8.2: SOTA Medical Named Entity Recognition as of Aug, 2019 on NCBI dataset. Source https://paperswithcode.com

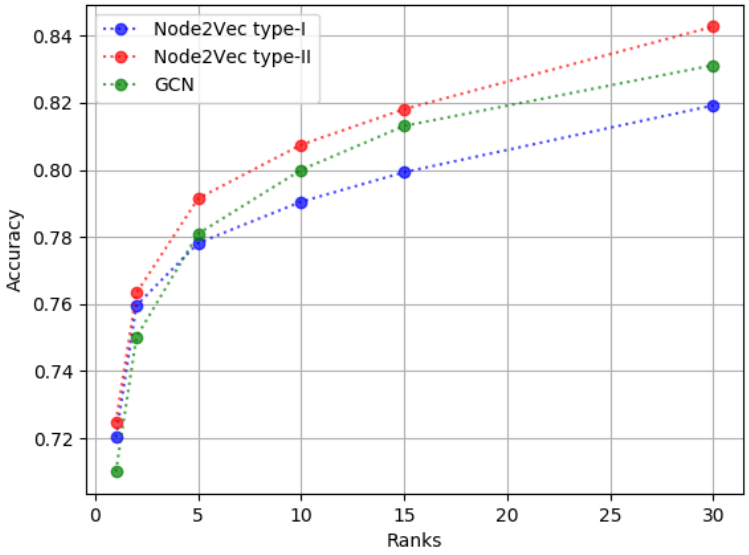


Figure 8.1: The plot of accuracy score at different rank of our disease normalization methods.