Neural Network Approaches to Medical Toponym Recognition

MohammadReza Davari

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

April 2020

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **MohammadReza Davari**

Entitled: **Neural Network Approaches to Medical Toponym Recognition**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. Charalambos Poullis

______________________ Examiner
Dr. Charalambos Poullis

______________________ Examiner
Dr. Tristan Glatard

______________________ Co-supervisor
Dr. Tien D. Bui

______________________ Co-supervisor
Dr. Leila Kosseim

Approved by ______________________
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

__________ 2020 ______________________
Dr. Amir Asif, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Neural Network Approaches to Medical Toponym Recognition

MohammadReza Davari

Toponym identification, or place name recognition, within epidemiology articles is a crucial task for phylogeographers, as it allows them to analyze the development, spread, and migration of viruses. Although, public databases, such as GenBank (Benson et al., November 2012), contain the geographical information, this information is typically restricted to country and state levels. In order to identify more fine-grained localization information, epidemiologists need to read relevant scientific articles and manually extract place name mentions.

In this thesis, we investigate the use of various neural network architectures and language representations to automatically segment and label toponyms within biomedical texts. We demonstrate how our language model based toponym recognizer relying on transformer architecture can achieve state-of-the-art performance. This model uses pre-trained BERT as the backbone and fine tunes on two domains of datasets (general articles and medical articles) in order to measure the generalizability of the approach and cross-domain transfer learning.

Using BERT as the backbone of the model, resulted in a large highly parameterized model (340M parameters). In order to obtain a light model architecture we experimented with parameter pruning techniques, specifically we experimented with Lottery Ticket Hypothesis (Frankle and Carbin, May 2019) (LTH), however as indicated by Frankle and Carbin (May 2019), their pruning technique does not scale well to highly parametrized models and loses stability. We proposed a novel technique to augment LTH in order to increase the scalability and stability of this technique to highly parametrized models such as BERT and tested our technique on toponym identification task.

The evaluation of the model was performed using a collection of 105 epidemiology articles from PubMed Central (Weissenbacher et al., June 2015). Our proposed model significantly improves the state-of-the-art model by achieving an F-measure of 90.85% compared to 89.13%.

Acknowledgments

I would like to thank Dr. Bui for his support and guidance.Under his supervision, I was able to explore many domains and applications of artificial intelligence and eventually find my passion. I would like to thank Dr. Kosseim for supporting my curiosities, while guiding me every step of the way. I am forever grateful for the care and motherly love she has shown to me throughout my studies. I would like to thank my lab colleague, Farhood Farahnak, for mentoring me. I learned a lot from our discussion in the lab and I truly enjoyed your company outside of it.

Lastly, I would like to dedicate this thesis to my family and girlfriend Jia Lu. Jia has been supportive of me and my dreams from the first day I have met her. Her valuable advices and her encouragements were instrumental in my studies, and for that I am forever thankful. My parents and my sister have always been the biggest fans of my work even if they had little idea of what exactly I am doing. Words cannot express my gratitude to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Phylogeographers, who study the geographic distribution of viruses, have long linked the increase in the geographical spread of viruses (Gautret et al., October 2012; Green and Roberts, November 2000) to the growth in global tourism and international trade of goods. Notable cases include, the 2006 outbreak of Escherichia coli O157:H7 across multiple states in the United States linked to spinach grown in California (Grant et al., October 2008). Due to cross border trade of goods, this local outbreak of Escherichia coli soon became pandemic and affected 20 states and infecting 205 people in the US (Grant et al., October 2008). In 2014, a few scattered cases of Ebola were first reported in Guinea; then shortly after the virus made its ways to Lagos, Nigeria (approximately 2000km from Guinea) as well as Dallas, Texas (approximately 9000km from Guinea) through air travel (World Health Organization, January 2015). This is shown in Figure 1.1.

Epidemiologists study and model the global impact of the spread of viruses by considering information on the DNA sequence and structure of viruses, but also by relying on accurate metadata. Although accurate localized geographical data is critical for their studies, most publicly available data sets, such as GenBank (Benson et al., November 2012), provide insufficient details on the matter, limited only to the country or state level. Therefore, a manual inspection of biomedical articles is vital in order to obtain more fine-grained localization information. Figure 1.2 shows an example of the process of modeling the global spread of ZIKV virus based on the epidemiology

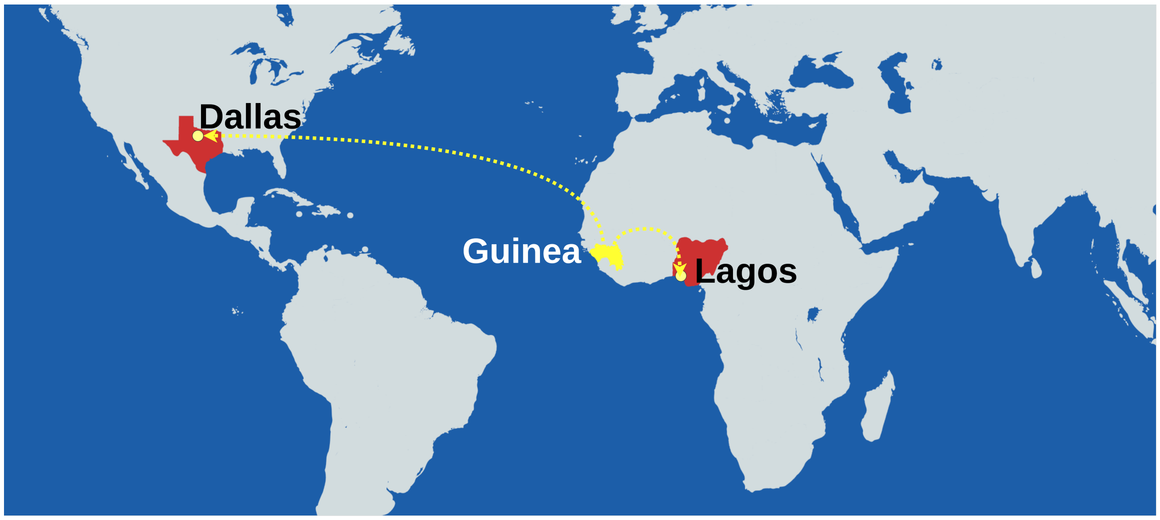
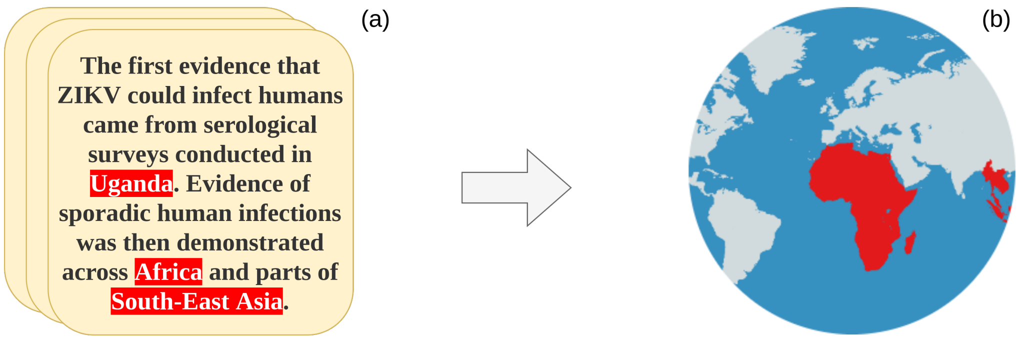


Figure 1.1: Outbreak of Ebola through air travel. Yellow: virus first seen, Red: Virus reported due to air travel.



The first evidence that ZIKV could infect humans came from serological surveys conducted in Uganda. Evidence of sporadic human infections was then demonstrated across Africa and parts of South-East Asia.

Figure 1.2: (a) Reports on the spread of ZIKV[1] (b) a model presenting the spread of the decease.

reports on the evidence of this virus. In this example, the report indicates that:

(1) The first evidence that ZIKV could infect humans came from serological surveys conducted in Uganda. Evidence of sporadic human infections was then demonstrated across Africa and parts of South-East Asia (Boeuf et al., August 2016).

Epidemiologists studying and trying to model the spread of the ZIKV virus, would need to manually extract the place names: *Uganda, Africa*, and *South-East Asia* to be able to model the global spread of this virus.

[1]This example is taken from (Boeuf et al., August 2016)

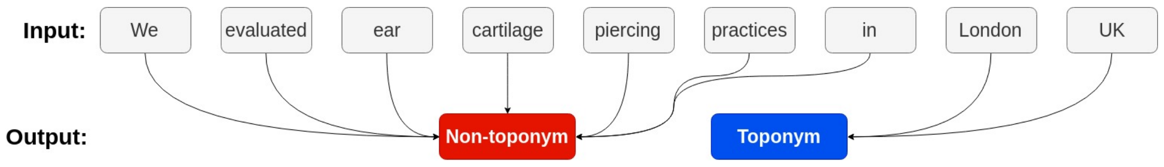


Figure 1.3: An example of input and expected output of toponym detection task.

Toponym resolution can be regarded as a specific application of Named Entity Recognition (NER), an active area of research in Natural Language Processing (NLP). NER addresses the problem of identifying and disambiguating phrases referring to entities (e.g. names of people, organizations, and geographic locations) in texts (Chiu and Nichols, July 2016; Collobert and Weston, July 2008; Lample et al., June 2016; Li et al., November 2015; Nadeau and Sekine, January 2007), while toponym resolution focuses only on names of geographic locations (Magge et al., July 2018). **Toponym resolution** hence refers to two problems: **toponym identification** and **toponym disambiguation**. Identifying the word boundaries of phrases that denote geographic expressions is the concern of toponym identification. For example, as shown in Figure 1.3, given the sentence:

(2) We evaluated ear cartilage piercing practices in London, UK.[2]

The task of **toponym detection** is to identify *London* and *UK* as toponyms, and all other words as non-toponym.

The task **toponym disambiguation** is to label each toponym with its geographic location. For example in Example 2, toponym disambiguation should map the detected toponyms (*London*, *UK*) to their corresponding geographical locations. For *London* alone, we have at least 11 choices of locations around world (e.g. [London, UK], [London, Ontario], [London, West Virginia]). In this example it is clear from the context of the sentence that the mention of *London* refers to the city in the *UK*.

In recent years, toponym resolution has been the subject of a number of studies (e.g. (Ardanuy and Sporleder, June 2017; DeLozier et al., February 2015; Taylor, December 2017)) which have demonstrated that the task is highly dependent on the textual domain (Amitay et al., July 2004; Purves et al., June 2007; Qin et al., November 2010; Kienreich et al., July 2006; Garbin and

[2]This example is taken from (Mandavia et al., June 2014).

Mani, October 2005). Previous methods used for toponym identification have relied on comprehensive gazetteers (Lieberman and Samet, December 2011) and hand crafted rules (Tamames and de Lorenzo, June 2010), which require significant work and expertise to adapt across domains. Hence, an automatic tool to detect and disambiguate toponyms for specific domains is necessary.

1.2 Goal of This Thesis

Motivated by recent research on the use of neural networks for NLP, the goal of this thesis is to experiment with neural approaches for toponym identification within the medical domain. Rather than relying on hand-crafting rules or on comprehensive gazetteers, this work investigates the use of architectures that not only automatically learn such rules and structures, but are also better predictors. In order to achieve this goal we used SemEval 2019 task 12 shared task (Weissenbacher et al., June 2019) dataset which contains 105 annotated bio-medical articles from PubMed. We approached the problem from two different angles:

(1) relying on transferred semantic information (i.e. word embeddings) coupled with specific linguistic insights (e.g. part of speech tags).

(2) relying on transferred knowledge from a comprehensive model of the language, allowing the language model to determine the needed linguistic features by itself.

1.3 Contributions

This thesis presents a number of contributions:

- a set of experiments evaluating the contribution of a variety of linguistic driven insights and embedding representations for toponym detection, in the medical domain. This gave rise to a paper at CICLing 2019 (Davari et al., April 2019).

- a novel approach to toponym detection in the medical domain, based on knowledge transfer from language models which achieves the state-of-the-art performance.

1.4 Thesis Structure

This chapter briefly defined the task of toponym resolution as two sub-tasks (detection and disambiguation) and motivated its importance within the medical domain. Given the drawbacks of previous conventional approaches, through this thesis, we investigated neural networks and language models for toponym identification. The rest of this thesis is structured as follows: Chapter 2 reviews the datasets used in our work, previous work on toponym identification, and the neural architectures used in Chapter 3 and 4. Chapter 3 presents and evaluates our first model: a feed forward neural network enriched with linguistic insights. Chapter 4 then expands the model developed in Chapter 3 with a language model based neural architecture for toponym identification and shows a significant improvement in performance. Finally Chapter 5 summarizes the thesis and proposes future work.

Chapter 2

Related Work

In Chapter 1, we briefly introduced the task of toponym identification: labeling each word of a text as toponym or non-toponym. Previous work on toponym detection can be categorized as:

- Non-Neural Approaches:
  (1) rule based approaches (e.g. (Tamames and de Lorenzo, June 2010))
  (2) dictionary or gazetteer-driven (e.g. (Lieberman and Samet, December 2011))
  (3) traditional machine learning approaches (e.g. (Santos et al., June 2015))
- Neural Approaches (e.g. (Magge et al., July 2018))

2.1 Non-Neural Approaches

The aim of rule based approaches is to manually record the contextual information and patterns that are indicative of the presence of toponyms. However, such indicative structures are limited and difficult to identify even by experts (Tamames and de Lorenzo, June 2010). Often, text samples sparsely manifest useful contextual information, and even if a pattern is correctly identified and captured, it may lead to false positive identifications, so its use should be evaluated. In addition, these handwritten rules are not capable of characterizing all possible cases, therefore leading to a number of false negative identifications.

Gazetteer based techniques (e.g. (Lieberman and Samet, December 2011)) rely on the existence of comprehensive databases of geographic names (e.g. GeoNames [1] and Google Maps [2]). These approaches allow to reach high levels of recall but suffer from a large number of false positive identification, resulting in a relatively low precision. This is because they are unable to correctly identify and disambiguate the entities that belong to multiple classes of NER. For example in the sentence,

(3) Alexander Hamilton was an American statesman and one of the Founding Fathers of the United States.[3]

the word *Hamilton* will be recognized as a location name since it exists within the database of geographic gazetteers; however, in the specific context of sentence (3), the entity is referring to a person. To combat the relatively large number of such in-context false positive identification, hand-written heuristics are typically used. However, defining heuristics requires accurate analysis of the corpus and expertise in the domain. While these heuristics improve the precision of the model, they decrease its generalizability, since these rules are mainly established from the patterns and statistics exhibited by the corpus used.

A dramatic shift in NER research occurred around 2015 with the advent of deep learning approaches. Along with the wider NLP community, NER research moved from traditional machine learning techniques to neural network approaches. By traditional machine learning techniques we refer to non-neural network approaches, including conditional random fields (CRFs), support vector machines (SVMs) and naive Bayes classifiers. Approaching toponym recognition via traditional machine learning techniques (e.g. (Santos et al., June 2015)) demands large, balanced, and accurately annotated corpora. Such quality datasets are often not available, hence leading to relative poor performance of this technique. Model training using this approach involves handcrafting representative features, which is a time consuming task and requires expert knowledge of the domain. Even with carefully engineered features, there is no guarantee that all relevant features have been modeled, hence the optimal performance of the method is highly dependent on the quality of the

[1] http://geonames.org
[2] https://www.google.com/maps
[3] This example is taken from https://en.wikipedia.org/wiki/Alexander_Hamilton.

engineered features.

2.2 Neural Approaches

State of the art approaches to NER (e.g. (Chiu and Nichols, July 2016; Collobert and Weston, July 2008; Lample et al., June 2016; Li et al., November 2015; Wang et al., November 2015)) are based on deep learning techniques. Compared to traditional machine learning approaches, deep learning techniques require relatively smaller datasets, as the knowledge gained from one task can be leveraged in another (Dai et al., June 2007; Wang et al., June 2016). Moreover, deep learning techniques are robust to label noise, and achieve outstanding generalization without the need for carefully annotated datasets (Rolnick et al., May 2017). These techniques learn to infer relevant features automatically leading to competitive performances and better predictive generalization.

The most commonly used architectures include: multi-layer perceptrons (MLP) (e.g. (Xu et al., July 2017)), convolutional neural networks (CNN) (e.g. (Collobert et al., August 2011)), and recurrent neural networks (RNN) (e.g (Chiu and Nichols, July 2016)). MLP and CNN architectures are used with the shared idea that only localized contextual information is needed for the prediction task. These methods are trained via defining a sliding contextual window and dismiss any contextual knowledge beyond the sliding window. MLP architectures are comprised of multiple layers of densely connected feed forward networks which allows for complex function approximation (Pal and Mitra, September 1992). CNN architectures reduce computational costs by taking advantage of mathematical cross-correlation and capturing reusable, transferable, and localized features. Although CNNs were originally designed as an architecture for computer vision tasks (LeCun et al., May 2010; Krizhevsky et al., December 2012; Oquab et al., June 2014), they have shown great ability to capture localized linguistic features and improving performance across a variety of tasks in NLP (Lopez and Kalita, March 2017; Mou et al., November 2016; Chen et al., August 2016). RNN architectures differ from CNN and MLP as they aim to take advantage of all available contextual information within a meaningful instance of data by using its internal state in subsequent processes of input sequences. In NLP, this meaningful structure is often the sentence, hence the RNN architecture tries to capture the structure and the contextual knowledge of an entire sentence

Table 2.1: Popular activation functions.

Name	Activation Function
Softmax	$\text{Softmax}(x)_i = \frac{\exp(x_i)}{\sum_i \exp(x_i)}$
Sigmoid	$\text{Sigmoid}(x) = \frac{1}{1+\exp(-x)}$
Hyperbolic tangent	$\tanh(x) = \frac{\exp(2x)-1}{\exp(2x)+1}$
Rectified Linear Units	$\text{ReLU}(x) = \max(x, 0)$

for its predictions.

2.3 Neural Building Blocks

In this section we will review the 3 neural architectures: feed-forward neural network (FFNN), convolutional neural network (CNN), and recurrent neural network (RNN). These networks are the building blocks of the more complex neural architectures used in this thesis (see Chapters 3 and 4).

2.3.1 Feed-Forward Neural Network

A feed-forward neural network is composed of one or many layers of fully or partially connected neural nodes which allows for complex function approximation (Pal and Mitra, September 1992). Each layer is composed of one or many nodes where each node represents a non-linear transformation function, the most popular of which are listed in Table 2.1.

Each layer receives as input a linear combination of the output of the previous layer. A non-linear transformation is then performed on these inputs to produce the output of the layer. More formally, let x be an n dimensional input vector of a layer (possibly from the output of the previous layer) containing m nodes, and f be a non-linear function, then the m dimensional output y is computed as:

$$y = f(Wx)$$

Where W is an $n \times m$ weight matrix learned during training. Figure 2.1 shows the architecture of a fully connected feed-forward neural network. For most NLP tasks, having a neural network

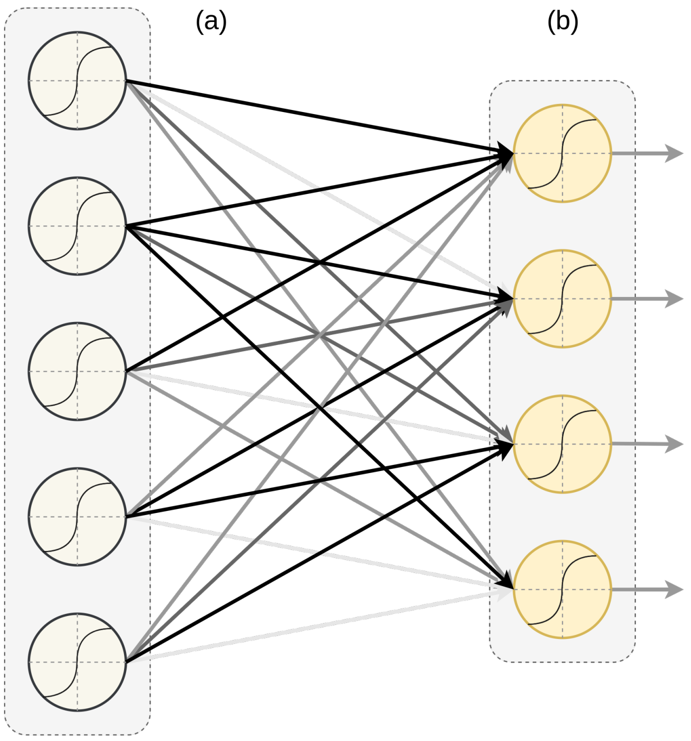


Figure 2.1: Feed-forward neural network: (a) Previous layer (b) Current fully connected layer.

architecture exclusively comprised of a feed-forward network is not optimal due to expensive computation cost of these networks and their inability to adjust to variable length input. The use of Convolutional (CNN) and Recurrent neural networks (RNN) mitigate these problems, which we will discuss further in Sections 2.3.2 and 2.3.3.

2.3.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) were first proposed as an architecture in the domain of computer vision and image processing (LeCun et al., October 1999). CNNs were developed with the assumption that certain features are shared across the input and it suffices to learn these features once and share them through the network. For example, in the context of computer vision, these shared features could be edges, colors, and shadows. Since these fundamental features exist within every portion of input images, the network can learn them by analysing each patch of the input. Patches of input are connected to neurons by performing convolution, and as the weights of the convolution matrix are shared, the network as a whole shares the knowledge. Figure 2.2 illustrates

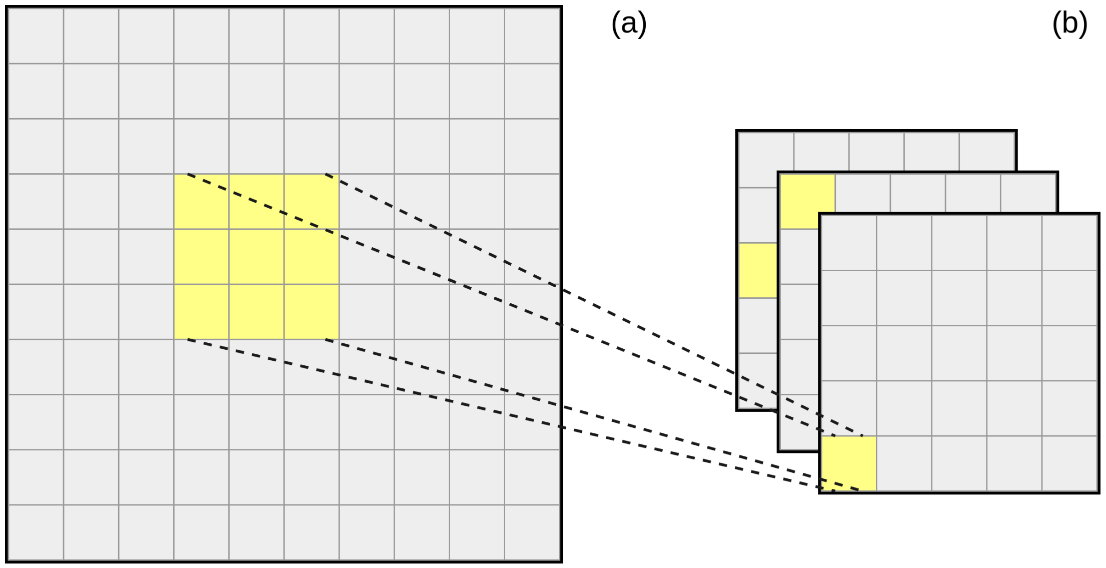


Figure 2.2: Convolutional Neural Network (CNN): (a) Input (b) Convolutional feature map applied on the input patches.

the convolution operation performed on an input.

Success of the CNNs in the image domain (LeCun et al., May 2010; Krizhevsky et al., December 2012; Oquab et al., June 2014) led to experimentation with this architecture for NLP tasks. CNNs have shown great ability to capture localized linguistic features and improve performance across a variety of tasks in NLP (Lopez and Kalita, March 2017; Mou et al., November 2016; Chen et al., August 2016). These networks significantly reduce the computation cost by learning shared features across the network. However, they are not well suited to deal with variable length inputs such as sentences. Recurrent neural networks are designed to remedy this issue, which we will discuss further in the following section.

2.3.3 Recurrent Neural Network

A recurrent Neural Network (RNN) is a natural extension of feed-forward neural network, allowing the layers to have connections to themselves in addition to the layers before and after them. This unique characteristic of the RNNs makes them the perfect candidate for the processing of variable length inputs since the design of the layers allows them to loop and consume the inputs. Furthermore, the intra-layer connections of RNN allows the network to capture and learn the sequential dependencies of the inputs, making them the preferred choice for any type of time series data (e.g.

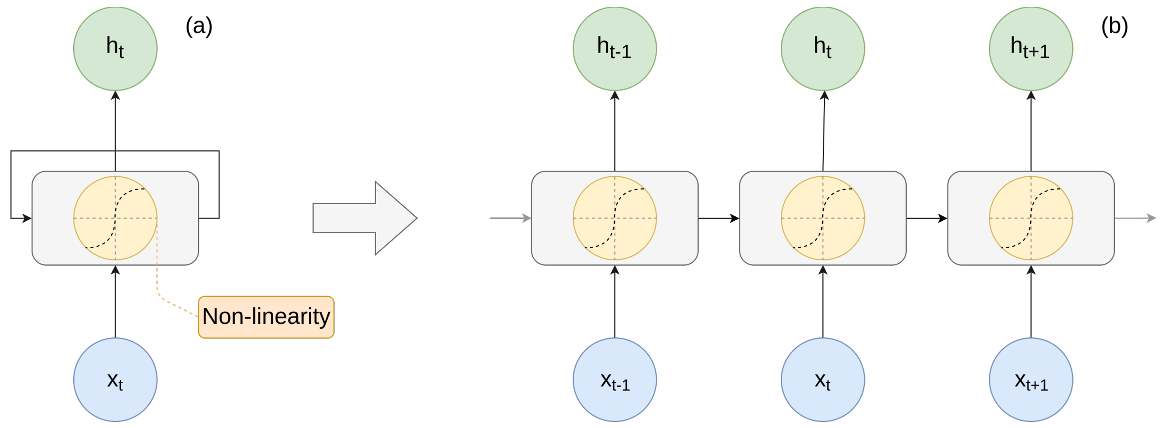


Figure 2.3: Recurrent neural network: (a) Recurrent network presented with the self-loop (b) Unrolled presentation of RNN with respect to time.

natural language text or speech). However, due to the sequential architecture of these network, training is not performed in parallel, leading to long training process for RNN based networks. Vaswani et al. (January 2017) proposed the Transformer architecture to mitigate this problem, which will be discussed in Section 2.7.

Due to the sensitivity of the network to the order of the input sequence, RNN networks are implemented as either a: **forward RNN** or a **backward RNN**. The only difference between these two types of RNNs, is the order in which the data is presented to the model. In the forward RNN, the data is presented to the network from the first element to the last, which in the backward RNN, the data is given to the network in reverse. For simplicity, from here on, when we refer to an RNN we mean the forward RNN.

In order to formally describe the RNN architecture, we let x be the input to the RNN, and h the output. For the time step t the network is defined as:

$$h_t = f(Ux_t + Wh_{t-1}) \tag{1}$$

where U is the matrix of weights connecting the input to the RNN unit, W is the matrix of weights used for the internal connections of the RNN, and f is some non-linearity (see Table 2.1). Figure 2.3 illustrates the general architecture of the RNN.

The most popular non-linearity for the RNN architecture (Equation 1) is the $\tanh$ function,

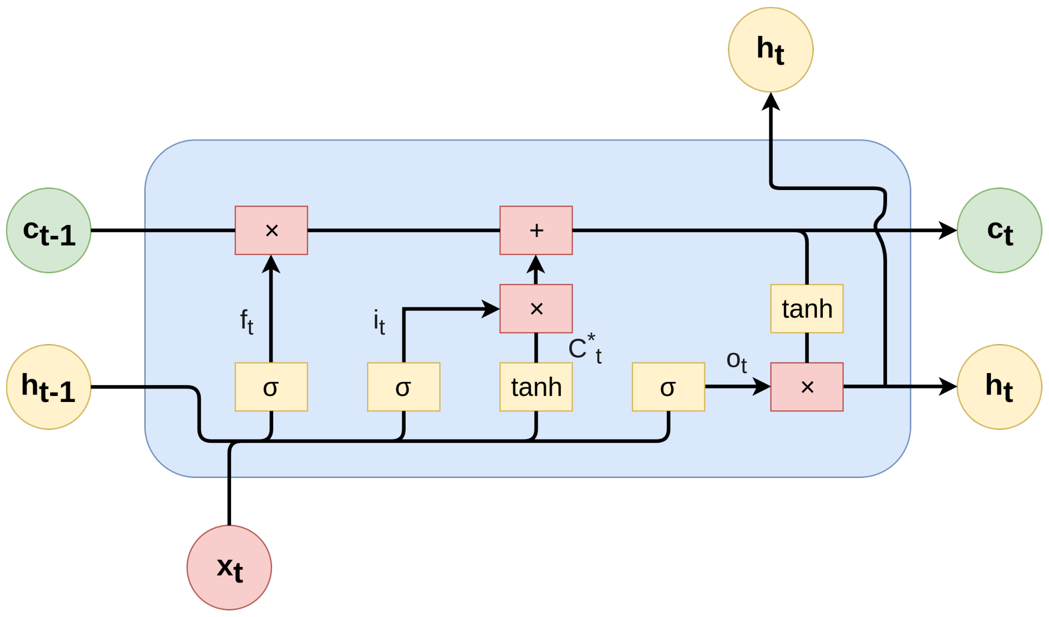


Figure 2.4: LSTM architecture.

leading to the so called “vanilla RNN”. Although RNNs are specialized in sequential inputs, the vanilla RNN is incapable of processing long inputs due to the vanishing and exploding gradient problems which prevents the system from learning (Bengio et al., March 1994; Hochreiter and Schmidhuber, November 1997). In order to mitigate this problem, 2 models were proposed: Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, November 1997) and Gated Recurrent Unit (GRU) (Cho et al., October 2014).

Long Short-Term Memory

Long Short-Term Memory (LSTM) was proposed by Hochreiter and Schmidhuber (November 1997) to mitigate the vanishing and exploding gradient problems (Bengio et al., March 1994). The proposed architecture augments the vanilla RNN with: an input gate, a forget gate, and an output gate. These gates allow the LSTM to learn to reset its sates when necessary. Figure 2.4 shows the architecture of an LSTM. In order to formally define LSTM, let the values of the input, forget, and output gates, be labeled by i, f, and o respectively. These values are computed at time t as:

$$\begin{aligned} i_t &= \sigma(U^i x_t + W^i h_{t-1}) \\ f_t &= \sigma(U^f x_t + W^f h_{t-1}) \\ o_t &= \sigma(U^o x_t + W^o h_{t-1}) \end{aligned}$$

where x_t is the input at time iteration t, h_{t-1} is the output of the LSTM unit at the previous time step, Us are the matrices of weights connecting the input to the LSTM unit, Ws are the matrices of weights used for the internal connections of the LSTM, and σ is the Sigmoid activation function. The final output of the LSTM at time t, h_t, is computed as:

$$h_t = \tanh(C_t) \odot o_t$$

Where C_t is called the *cell state* and is computed via the following 2 equations:

$$\begin{aligned} C_t^* &= \tanh(U^g x_t + W^g h_{t-1}) \\ C_t &= \sigma(f_t \odot C_{t-1} + i_t \odot C_t^*) \end{aligned}$$

Although the LSTM is more robust towards the vanishing and exploding gradient problem (Hochreiter and Schmidhuber, November 1997), and performs relatively better when it is presented with longer length inputs (Sutskever et al., December 2014), the additional parameters make it expensive to train. GRUs (Cho et al., October 2014) were introduced as an alternative to LSTMs in order to reduce the computation cost of the network while preserving the robustness of the model when presented with longer sentences.

Gated Recurrent Unit

As mentioned in the previous section, GRUs (Cho et al., October 2014) were developed as an alternative to LSTMs in order to reduce the computation cost, while preserving the robustness of the model. GRUs augment vanilla RNNs with only two gates: an update and a reset gate. Figure 2.5

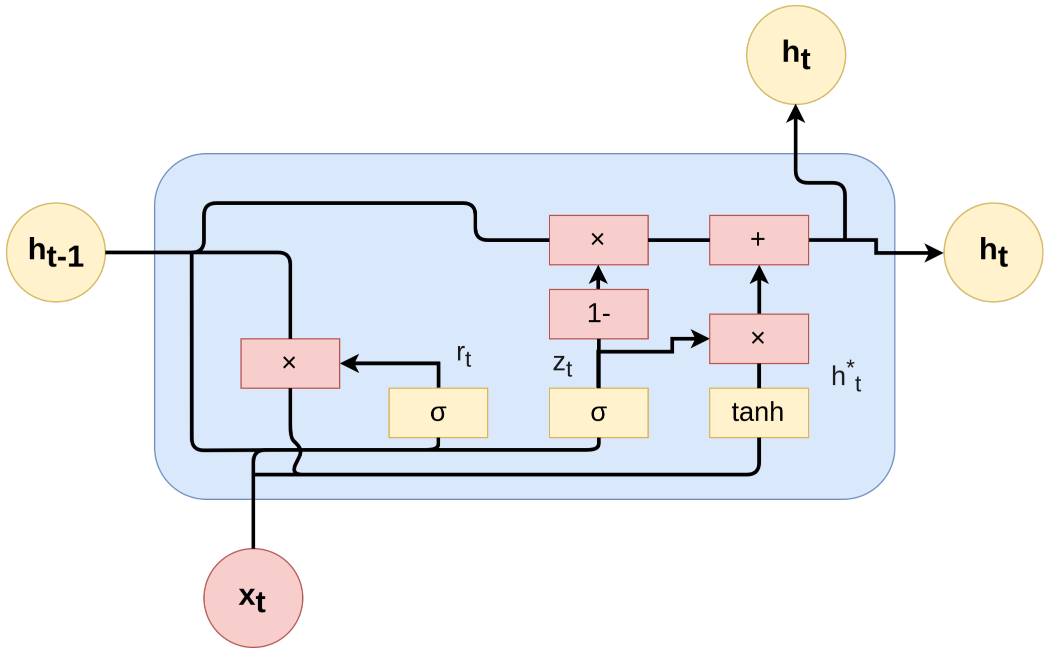


Figure 2.5: GRU architecture.

illustrates the architecture of GRUs.

In order to formally define GRUs, let z, r, and h^* represent the value of the update gate, reset gate, and the internal memory of GRU. Given input x at time step t, we have:

$$\begin{aligned} z_t &= \sigma(U^z x_t + W^z h_{t-1}) \\ r_t &= \sigma(U^r x_t + W^r h_{t-1}) \\ h^*_t &= \tanh(U^h x_t + (r_t \odot W^h h_{t-1})) \end{aligned}$$

Where Us are the matrices of weights connecting the input to the GRU unit, Ws are the matrices of weights used for the internal connections of the GRU. The final output of the GRU is computed as:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h^*_t$$

Table 2.2: Statistics of the SemEval 2019 task 12 shared task (Weissenbacher et al., June 2019) dataset.

	Training	Development	Test	Total
Size	2.8MB	0.5MB	1.5MB	4.8MB
Number of articles	63	10	32	105
Average size of articles (in words)	6422	5191	6146	6220
Average toponyms per article	43	44	50	45

2.4 Toponym Resolution in The Epidemiology Domain

Toponym resolution in the epidemiology domain is a relatively new research area. Previous attempts at developing an accurate toponym detector in this domain includes rule based approach (Weissenbacher et al., June 2015), Conditional Random Fields (Weissenbacher et al., November 2017), and a mixture of deep learning and rule based approaches (Magge et al., July 2018). However, most recent work in the area has been done within the context of the SemEval 2019 shared task 12 (Weissenbacher et al., June 2019).

In order to provide a common comparison point, the shared task organizers of the SemEval 2019 task 12 (Weissenbacher et al., June 2019) provided a base model. This baseline uses the Deep Feed Forward Neural Network (DFFNN) architecture of (Magge et al., July 2018) and is composed of 2 hidden layers with 150 rectified linear unit (ReLU) activation functions per layer, and a Softmax output layer. The baseline model is reported to have an F1 score of 69.84% with the dataset provided.

This dataset contains 105 articles from PubMed annotated with toponym mentions and their corresponding geographical locations. The dataset was split into 3 subsets: training, development, and test set containing 60%, 10%, and 30% of the dataset respectively. Throughout this thesis the same subsets were used to train, fine tune, and evaluate the performance of the models. More detailed statistics of the dataset are presented in Table 2.2. The evaluation of the models presented in this thesis has been done with the SemEval 2019 task 12 shared task (Weissenbacher et al., June 2019). Therefore, the training and performance evaluation of the models are performed using the dataset and the scorer function[4] provided by the organisers.

[4] https://competitions.codalab.org/competitions/19948#learn_the_details-evaluation

2.5 Metrics

The standard metrics for the task of toponym detection are: precision, recall and F-measure. These metrics can be measured in two ways: strict or overlapping measures. The strict measures, consider that a prediction to match with the gold standard annotation if both point to the exact same span of text. On the other hand, the overlapping measures, are more lenient as they consider a prediction to match with the gold standard annotations when they share a common span of text. The leniency of this measure depends on the size of the overlapping common span of the text between the predictions and the gold standard annotations.

(4) San Diego is a city on the Pacific coast of California.

If the system only identifies *Diego* as toponym, the overlapping measure counts this prediction as a success since it shares one common token with *San Diego*, a toponym. On the other hand, the strict measure would count it as a fail, since the whole *San Diego* was not predicted as toponym.

Since the research community in toponym identification is more concerned with strict measures (Magge et al., July 2018), we will only report on the strict measures of precision, recall and F-measure. We will compute the precision and recall for toponym detection using the standard equations:

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP+FP} \\ \text{Recall} &= \frac{TP}{TP+FN} \end{aligned}$$

where TP (True Positive) is the number of toponyms correctly identified by a toponym detector in the corpus, FP (False Positive) the number of phrases incorrectly identified as toponyms by the detector, and FN (False Negative) the number of toponyms not identified by the detector. As the definition of the precision and recall entails, optimization of one measure, independent of the other can lead to poor performance on the other measure. In order to have a single measure to represent precision and recall, the F-measure was developed as:

$$\text{F}\alpha \quad = \quad (1+\alpha^2) \times \frac{\text{Precision} \times \text{Recall}}{(\alpha^2 \times \text{Precision}) + \text{Recall}}$$

where the parameter α indicates an importance of precision compared to recall. In order to have both precision and recall to have equal weights the F1-measure is used:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.6 Attention Mechanisms

2.6.1 Motivation

Attention mechanisms in the topic of learning algorithms are motivated by how we, humans, pay attention to different components of our sensory inputs. In the context of visual attention, depending on our objective, we bring different components of our visual input to focus and blur the rest (Hoffman and Subramaniam, January 1995). Figure 2.6 (a) shows an image of maple taffies with a patch of the image masked. If we were to guess the content of the masked region, we would pay more attention to certain areas of the image, while blurring out the rest of the regions. The pink areas shown in Figure 2.6 (b), depicting the twisted fingers holding a popsicle stick, some maple syrup on the snow, and a popsicle stick attached to the maple syrup, will lead us to guess that the masked region must be covering a rolled up maple taffy. Other regions in Figure 2.6 (b) such as the background or the color of the person's sleeve (indicated by gray circles) do not contribute to our decision making.

In the context of natural language, we perceive similar contextual correlation between different components. For example in the sentence:

(5) I drank a glass of water.

We expect a liquid to appear in the sentence once we read the word *drank*. There is a strong correlation between these two words in this sentence. Hence, as shown in Figure 2.7, the word *drank* attends to the word *water*, however it does not directly attend to the word *glass*.

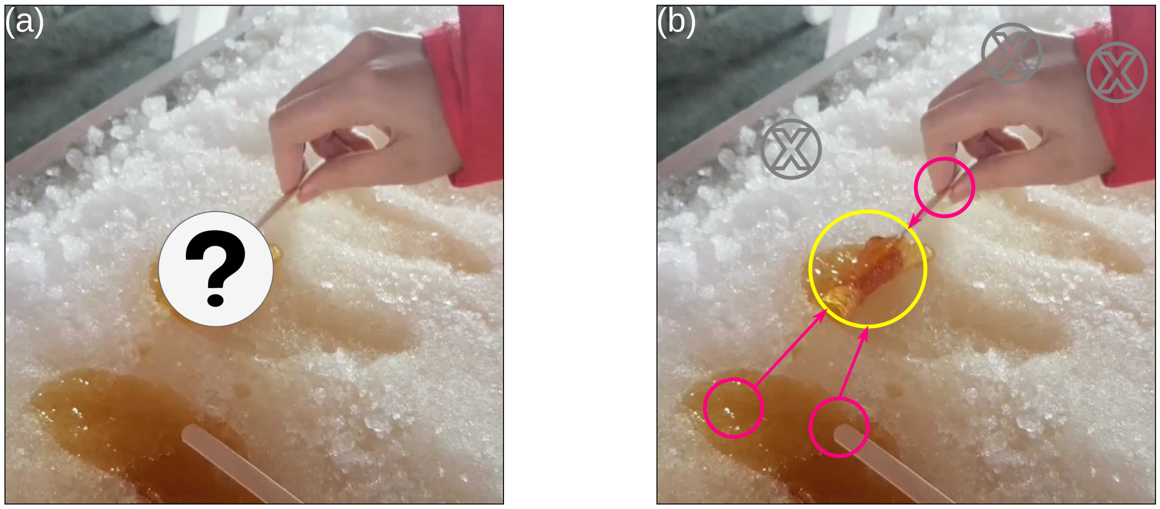


Figure 2.6: (a) Masked image (b) Attention to the pink circles help guess the content of the masked region, while the gray regions receive no attention.

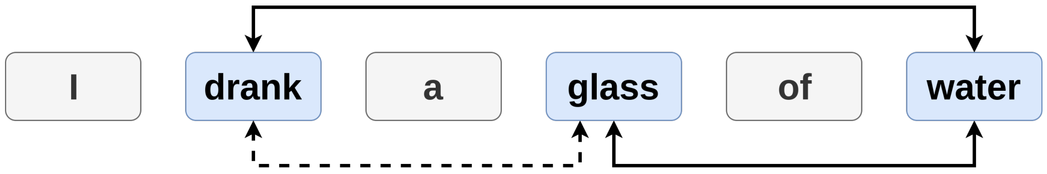


Figure 2.7: Solid arrows indicate high attention. Dashed arrow indicate low attention.

In the context of learning algorithms, attention is a mechanism that distributes importance weights to the components of an input (e.g. pixels in the image domain and words in natural language) in order to infer a target output. These importance weights indicate the correlation between the input components and the target output or in other words they specify how strong the algorithm should attend to different components of the input to infer the target output.

2.6.2 Origin of Attention Mechanism

In order to better understand the importance and advantages of attention mechanism, we first need to look at the problem it tries to solve. For this purpose, we briefly examine the sequence to sequence model architecture.

The sequence to sequence model or encoding-decoding architecture is an extension of the RNN. It is the standard model architecture for many NLP tasks such as: language modeling (Sutskever et al., December 2014), neural machine translation (Bahdanau et al., May 2015; Cho et al., October 2014), and syntactic constituency parsing (Vinyals et al., December 2015). This architecture, transforms an input or source sequence to an output or target sequence. These sequences can be of arbitrary length and not necessarily equal to each other. The architecture of sequence to sequence model is composed of: an **encoder mechanism** and a **decoder mechanism**.

The encoder operates on the source sentence, and compresses it to a fixed length vector known as the context vector or sentence embedding. The context vector is expected to be a rich representation of the source sentence containing a sufficient summary of the source information. A classical choice for the context vector is the last hidden state of the encoder (Cho et al., October 2014). The decoder constructs the target sentence based on the context vector it receives from the encoder. Both encoder and decoder architectures are based on RNNs i.e. using LSTM or GRU units (see Section 2.3.3). Figure 2.8 shows the encoder-decoder model used in neural machine translation for the following translation:

(6) English: Watermelon is delicious.
French: La pastèque est délicieuse.

Bahdanau et al. (May 2015) showed that a major drawback of using fixed-size context vector

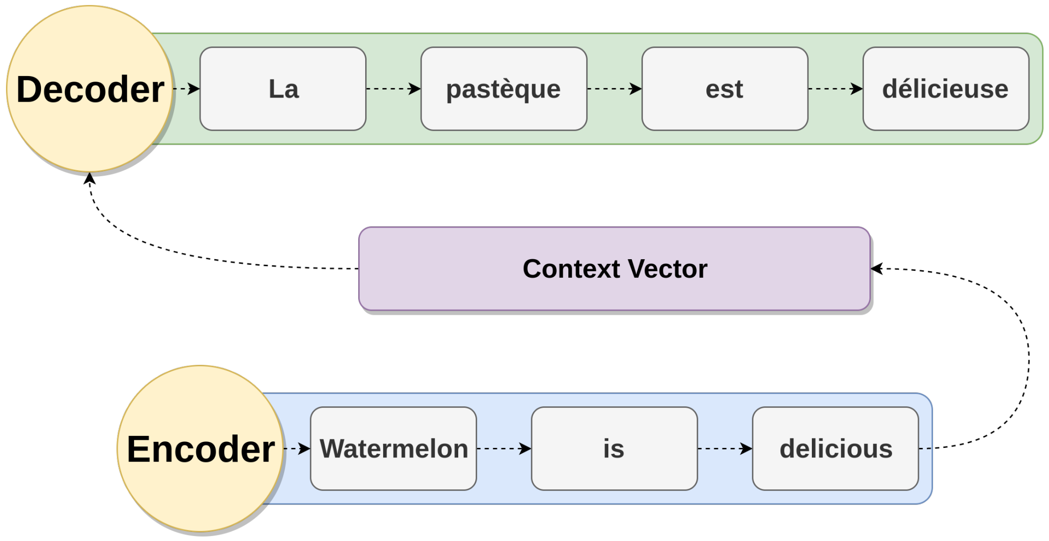
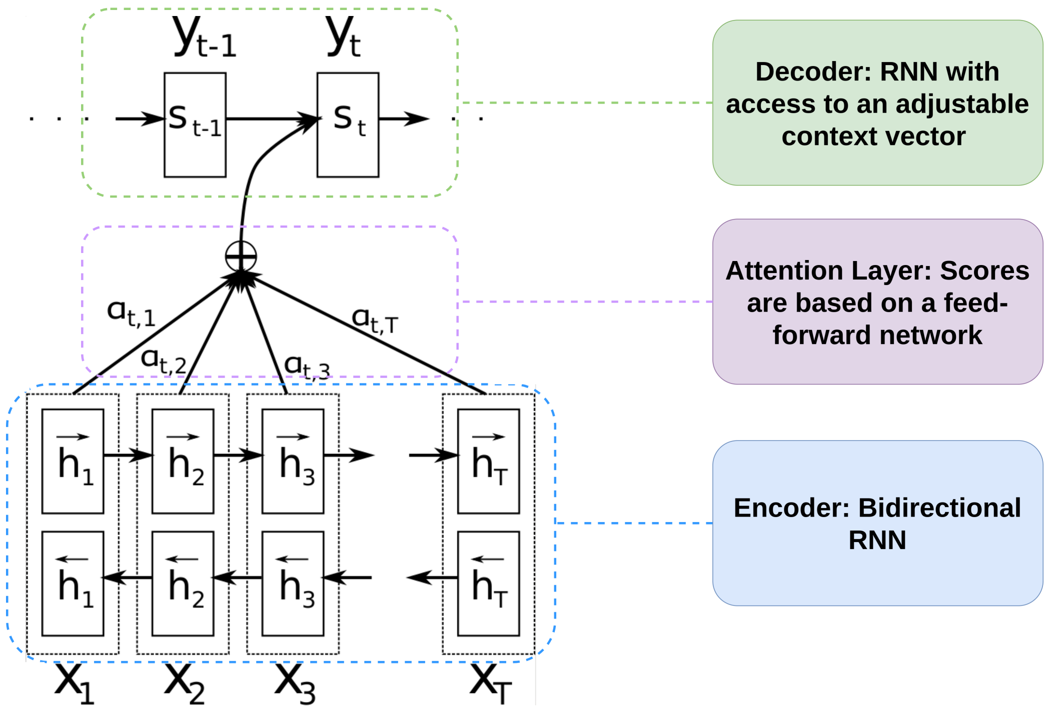


Figure 2.8: Encoder-decoder architecture used in neural machine translation, translating the sentence *Watermelon is delicious* to French.

is the limitation of this vector to summarize and remember all the necessary information in the source sentence. Having a fixed-size context vector introduced a bottleneck on the performance of sequence to sequence models. When the model is presented with longer length source sentences, the model would simply forget some of the information from the earlier part of the source sentence. In the context of neural machine translation, this led to poor and incoherent translations for longer sentences (Bahdanau et al., May 2015). Attention mechanism was, therefore proposed by Bahdanau et al. (May 2015) to remedy this issue.

In the context of Neural Machine Translation (NMT), Attention mechanism helps the encoder-decoder network memorize longer length source sentences. Attention mechanism allows the context vector to create links between the entire hidden representations of the source sentence, instead of using a single fixed sized context vector from the last hidden state of the encoder. These links are parameters learned by the network and they are adjusted for each output element in the target sequence. Since the context vector has access to the entire source sentence, the performance of the encoder-decoder network is not affected by the length of the source sentences.



y_{t-1}
y_t
s_{t-1}
s_t
a_{t,1}
a_{t,2}
a_{t,3}
a_{t,T}
h_1
h_2
h_3
h_T
x_1
x_2
x_3
x_T
Decoder: RNN with access to an adjustable context vector
Attention Layer: Scores are based on a feed-forward network
Encoder: Bidirectional RNN

Figure 2.9: Neural machine translation architecture used by Bahdanau et al. (May 2015)[5].

2.6.3 Formal Definition

Since the attention mechanism was introduced in NMT, we will base the examples of this section on this task, and we will focus on the encoder-decoder architecture that was proposed by Bahdanau et al. (May 2015). Assume, that we have a source sequence x of length T and the target sequence y of length M:

$$\begin{aligned} x &= [x_1, x_2, \ldots, x_T] \\ y &= [y_1, y_2, \ldots, y_M] \end{aligned}$$

The encoder will receive the source sequence x and will produce hidden state representations h_i at time step i. As shown in Figure 2.9, in the architecture proposed by Bahdanau et al. (May 2015) the encoder is a bidirectional RNN and h at time i is defined as:

$$h_i = \left[\overrightarrow{h_i}, \overleftarrow{h_i}\right] \;\; \forall i \in \{1, 2, \ldots, T\}$$

Where $\overrightarrow{h_i}$ is the hidden state representations in the forward pass of the RNN and $\overleftarrow{h_i}$ is the hidden state representations in the backward pass of the RNN. The decoder will produce hidden state representations s_j defined for time j as:

$$s_j = f(s_{j-1}, y_{j-1}, c_j) \;\; \forall i \in \{1, 2, \ldots, M\}$$

Where f computes the current hidden state given the previous hidden state, the previous output, and the context vector. f can be either a vanilla RNN unit, a GRU, or an LSTM unit. The parameter c_j is the context vector at time j computed as a weighted sum of the source sequence hidden representations:

$$c_j = \sum_{i=1}^{T} \alpha_{ji} h_i \tag{2}$$

Where the weights α_{ji} for each source sequence hidden state representation, are alignment measures indicating how well an input at position i and an output at position j match:

$$\alpha_{ji} = \text{align}(y_j, x_i) \tag{3}$$

The alignment measure is a probability distribution over a predefined alignment score function. The score for the input at position i and output at position j is computed based on the hidden representation of the input at position i, h_i, and the hidden representation of the decoder at position $j-1$, right before emitting the output y_i:

$$\text{align}(y_j, x_i) = \frac{\exp\left(\text{score}(s_{j-1}, h_i)\right)}{\sum_{r=1}^{T} \exp\left(\text{score}(s_{j-1}, h_r)\right)}$$

In the architecture proposed by Bahdanau et al. (May 2015) a feed-forward neural network is used to parametrize and learn the alignment scores. The feed-forward network is composed of a single

[5] Source of figure (Bahdanau et al., May 2015)

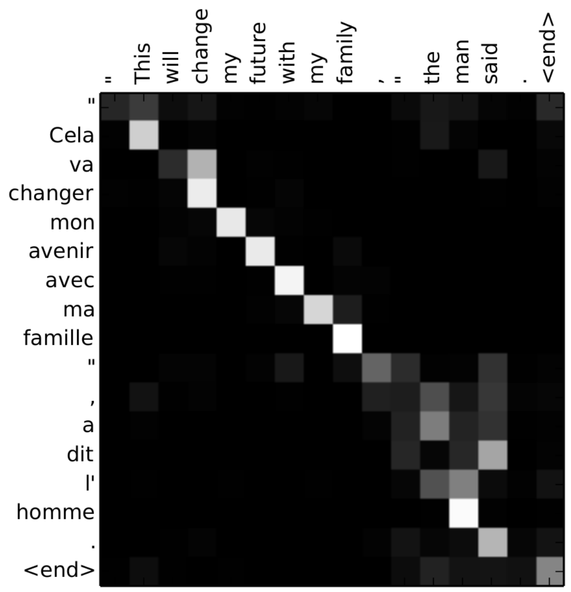


Figure 2.10: Matrix of alignment scores for the translation of *“This will change my future with my family,” the man said.* to French, *“Cela va changer mon avenir avec ma famille”, a dit l’homme.*[6]

hidden layer with tanh activation function and is jointly trained with the other parts of the network. Hence, the alignment scores are given by:

$$\text{score}(s_j, h_i) = v \tanh\left(W[s_j, h_i]\right)$$

Where v and W are weight matrices that will be learned by the network. These alignment scores define how much of each of the source hidden states is needed to produce each of the target outputs or in other words, how much the target words should attend to the source sequence in the decoding process. This concept is captured by the matrix of the alignment scores, explicitly showing the correlation between input and output words. Figure 2.10 shows the matrix of alignment scores for an English-French translation.

2.6.4 Variations of Attention Mechanism

Success of the attention mechanism in NMT motivated researchers to use it in different domains (e.g. computer vision (Xu et al., July 2015)) and experiment with various forms of this

[6]Source of figure (Bahdanau et al., May 2015)

Table 2.3: Popular Alignment Score Functions for an Attention Mechanism.

Name	**Alignment Score Function**	**Used In**
Content-based	$\text{score}(s_j, h_i) = \cos(s_j, h_i)$	Graves et al. (December 2014)
Additive	$\text{score}(s_j, h_i) = v\tanh\left(W[s_j, h_i]\right)$	Bahdanau et al. (May 2015)
Dot Product	$\text{score}(s_j, h_i) = s_j^T h_i$	Luong et al. (September 2015)
General	$\text{score}(s_j, h_i) = s_j^T W h_i$	Luong et al. (September 2015)
Location-base	$\text{score}(s_j, h_i) = W h_i$	Luong et al. (September 2015)
Scaled Dot Product	$\text{score}(s_j, h_i) = \frac{s_j^T h_i}{\lvert\lvert h_i \rvert\rvert}$	Vaswani et al. (January 2017)

mechanism (Vaswani et al., January 2017; Luong et al., September 2015; Britz et al., September 2017). The first natural extension to this mechanism is the alignment score function.

As discussed in Section 2.6.3, Bahdanau et al. (May 2015) used a single feed-forward neural network with a $\tanh$ activation function to compute the alignment scores. However other approaches have been proposed for the alignment score function. Table 2.3 lists a few popular alignment score functions.

Aimed to reduce the computation costs of attention mechanism, Xu et al. (July 2015) experimented with two kinds of attention mechanism: **soft attention** and **hard attention**. Soft attention is similar to the attention mechanism introduced by Bahdanau et al. (May 2015) as it assigns a (soft) probability distribution to all the source hidden states, which makes the model smooth and differential but costly in the computation time.

On the other hand , hard attention aims to reduce the computation cost of attention mechanisms by only focusing on one single source hidden representation at a time. The attention mechanism in this setting is representing a multinoulli probability distribution over all the source hidden states. Therefore, the vector of the attention weights is a one-hot vector assigning a weight of 1 to the most relevant source hidden state and 0 to the others.

The one-hot representation of the attention is non-differentiable hence it requires more complicated techniques such as variance reduction or reinforcement learning to train (Luong et al., September 2015). In order to remedy the non-differentiability of hard attention, Luong et al. (September 2015) proposed the concept of **local attention**. In their work, they call the soft attention mechanism, the **global attention** since it attends to all hidden states in the source sequence. The local attention, on the other hand, only attends to a window of the source hidden states. This mechanism first predicts a single aligned position for the current target word mimicking the behavior of the hard attention. A window centered around the source position is then used to compute the context vector similar to the mechanism of soft attention. The local attention mechanism perfectly blends soft and hard attention together to save computation costs while preserving the differentiability of the model.

Self-attention or intra-attention is a special case of the attention mechanism where the source and target sequence are the same sequence. The context vector formulation is the same as in Equation 2, however, the weights are formulated differently. As a result, the target sequence in Equation 3 is replaced by the source sequence leading to:

$$\alpha_{ji} = \text{align}(x_j, x_i)$$

The attention mechanism in this setting will find the best correlation between each word in a sentence and the others, making self-attention an integral part of the recent advancements in embedding representations (Vaswani et al., January 2017; Devlin et al., June 2019; Yang et al., December 2019).

2.7 Transformer

In Section 2.3.3 we introduced RNNs. Due to their ability of processing sequential inputs of variable length, these architectures have been the preferred building block for many NLP neural approaches such as language modeling (Sutskever et al., December 2014), neural machine translation (Bahdanau et al., May 2015; Cho et al., October 2014), and syntactic constituency parsing (Vinyals et al., December 2015). However, RNNs are only slightly parallelizable, that means the computational resources cannot be fully utilized during training and hence, leading to a very time consuming training process.

In order to mitigate this issue, Vaswani et al. (January 2017) proposed the Transformer architecture. The Transformer model is solely based on the attention mechanism (see Section 2.6) and uses self attention layers to learn word representations. In the context of sequential data, the Transformer architecture is superior to the classical neural architecture approaches such as RNNs or CNNs based on three important criteria: computation complexity, parallelizability, and long term dependency modeling.

The computation complexity of Transformer models is $O(n^2.d)$ for a sequence of length n

and hidden representation of size d, as opposed to RNNs and CNNs which have a computation complexity of $O(n.d^2)$ and $O(k.n.d^2)$ respectively, where k is the kernel size of the convolution. The dominating factor determining computation complexity of the model is the dimension of the hidden representation, since it is typically far larger than the sequence length or the kernel size. Hence, the Transformer model is conserving computation complexity by $O(d)$ compared to the other two models.

As mentioned in Section 2.3.3, RNN computations are only slightly parallelizable, leading to a sequential computation of $O(n)$ on a sequence of size n, since the model essentially needs to loop through the sequence. However, Transformer and CNN models are highly parallelizable by design, having $O(1)$ sequential computations.

Modeling long term dependencies of a sequence input is a challenging task (Bengio et al., March 1994; Bahdanau et al., May 2015). The length of the path between long range dependencies has an inverse correlation with the ability of the model in learning these dependencies. Longer paths prevent the gradient or learning signals to be transmitted smoothly (Bengio et al., March 1994). Hence the shorter the path between long range dependencies, the better the model learns. CNNs with a kernel of size k have a maximum path length of $O(\log_k(n))$ for a sequence of size n, while RNNs have a maximum path length of $O(n)$. Since Transformers are solely based on attention mechanism, the maximum path length in this architecture is $O(1)$, letting the model to seamlessly capture long term dependencies of sequential inputs.

2.7.1 Multi-Head Attention

Vaswani et al. (January 2017) introduced the multi-head attention mechanism in order to jointly attend to information from different representation sub-spaces at different positions. Rather than only computing the attention once, the multi-head attention mechanism independently attends to the source information multiple times in parallel and then concatenates the results to provide a richer representation of the source sequence. This allows the attention model to capture different kinds of dependencies within the source sequence such as: semantic dependencies, syntactic dependencies, and grammatical gender dependencies. Figure 2.11 shows the different types of dependencies captured via 8 attention heads for the word *because* in the sentence *The animal didn't cross the street*

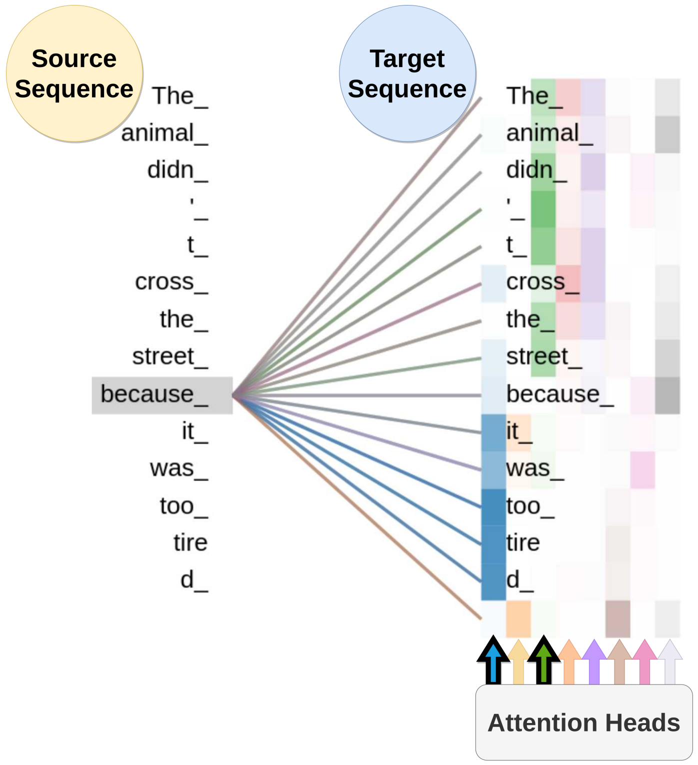


Figure 2.11: Matrix of alignment scores of the multi-head self attention model for the word *because* in the sentence *The animal didn't cross the street because it was too tired.*[7]

because it was too tired. In particular, we will focus on the contingency dependency in this figure. The word *because* is an explicit discourse marker which indicates a contingency relation. The blue and green attention heads (marked with thicker borders) in Figure 2.11 have successfully captured this dependency relation.

The scaled dot product attention is used in all instances of the attention mechanism in the Transformer model, since it can be implemented using highly optimized matrix multiplication algorithms. Transformer views the encoded representation as key-value pairs (K, V) of dimension n, although both the keys and values are the encoder hidden states, this distinction in notation helps with better understanding of the model. The output of the decoder is represented by Q, the query, of size m. The attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{n}}\right)V$$

[7]The image was produced using the pre-trained Transformer via Tensor2tensor (Vaswani et al., March 2018)

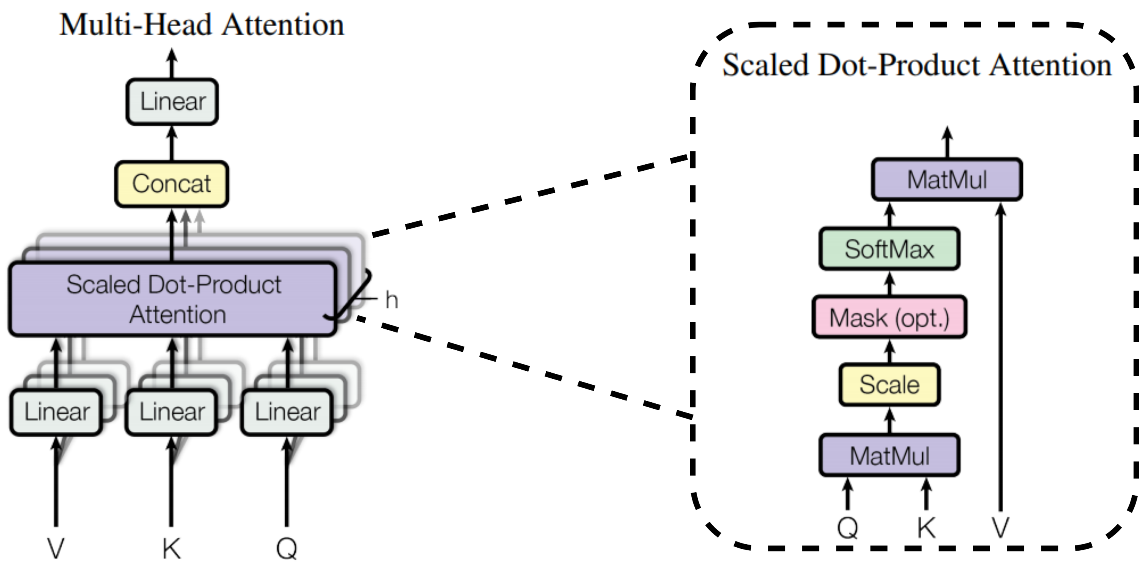


Figure 2.12: Multi-head attention architecture.[8]

The multi-head attention with h heads performs the above operation h times, then concatenates the outputs and performs a linear transformation for the final result, given as:

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= [\text{head}_1,\ldots,\text{head}_h]W^O \\ \text{where}\ \ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

Where W^O, W_i^Q, W_i^K, and W_i^V are matrix projections to be learned. Figure 2.12 shows the multi-head attention architecture.

2.7.2 Model Architecture

The Transformer model was developed specifically for the NMT and follows the same principles of the sequence to sequence models (Sutskever et al., December 2014). The model is comprised of two modules: the **encoder** and the **decoder** module.

The encoder module (shown in the left side of Figure 2.13) generates an attention-based representation. It consists of a stack of 6 identical layers, where each layer is composed of 2 sub-layers: a multi-head attention layer and a position-wise fully connected feed-forward network. In order to

[8]The image was taken from (Vaswani et al., January 2017)

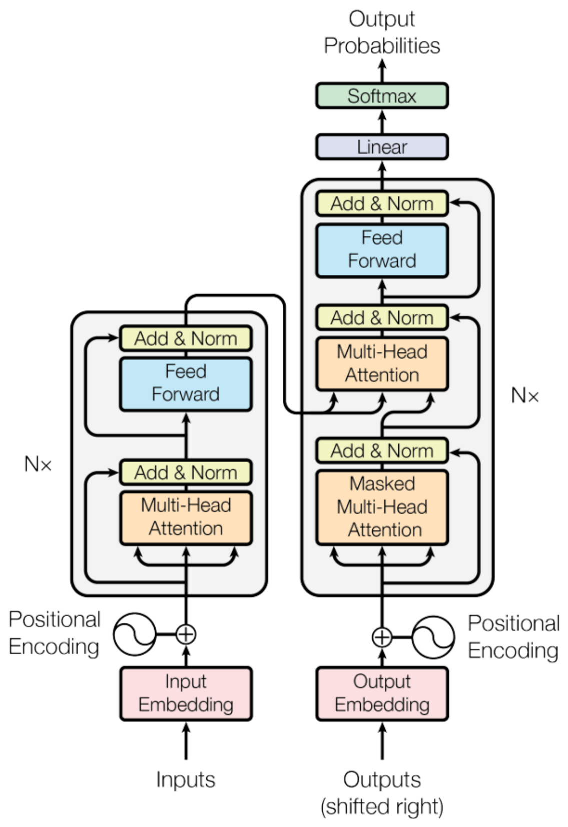


Figure 2.13: Transformer model architecture.[9]

encourage gradient flow in each sub-layer, a residual connection (He et al., June 2016) is formed followed by a normalization layer (Ba et al., July 2016), i.e. the output of each sub-layer is given by:

$$\mathrm{Output} = \mathrm{LayerNorm}\left(x + \mathrm{Sublayer}(x)\right)$$

Where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself.

The decoder module (see the right side of Figure 2.13) also consists of a stack of 6 identical layers. Similar to the encoder, each layer is composed of sub-layer, in addition to the two sub-layers in each encoder layer, the decoder incorporates a third sub-layer, which performs multi-head attention over the output of the encoder stack. Analogous to the encoder module, each sub-layer adopts a residual connection and a layer normalization. The first multi-head attention sub-layer of the decoder module is modified with a masking mechanism, in order to prevent the decoder to look into the future.

[9]The image was taken from (Vaswani et al., January 2017)

The Transformer model does not contain any recurrence or convolution, hence the model is unaware of the order of the sequence. In order to augment the model with this information a positional encoding is added to the input embeddings. The positional encoding captures the relative positioning of the elements of the sequence via sine and cosine functions of different frequencies. The i^{th} dimension of a positional encoding of size n for the j^{th} position is defined as:

$$\begin{aligned}\text{PositionalEncoding}_{(j,2i)} &= \sin\left(\frac{j}{10000^{\frac{2i}{n}}}\right)\\ \text{PositionalEncoding}_{(j,2i+1)} &= \cos\left(\frac{j}{10000^{\frac{2i}{n}}}\right)\end{aligned}$$

2.8 Tokenization

Classical approaches to tokenization in English rely on punctuation between words to constitute token boundaries. This method of segmentation does not generalize well to languages where punctuation between words does not exist or exists but on a very limited scale (e.g. Chinese, Japanese, and Korean). Word representations obtained from this naive approach to segmentation result in a large vocabulary that is domain specific, and it is unable to handle out-of-vocabulary (OOV) words.

Intuitively rare and unknown words can often be decomposed into multiple representative and meaningful sub-word units. For example, morphologically complex words can easily be defined by the sum of their morphemes. This idea is the main motivation behind sub-word tokenization, which leads to a compact network vocabulary and improves cross domain generalizability (Sennrich et al., August 2016).

Embeddings based on sub-word tokenization such as BERT (see Section 2.10.2) assign unique vector representations for more frequent words, whereas less frequent words will be decomposed into and represented by sub-word units that best retain their meaning. For example, a conventional word embedding model (Mikolov et al., May 2013) would learn four different vector representations for the words `[high, higher, low, lower]`, but embeddings based on sub-word tokenization could attain smaller neural vocabulary by taking advantage of the compositionality of the language. In this example we could have three vectors for `[high, low, er]` and represent every word in our corpus as a linear combination of these three vectors.

In this section we will briefly discuss two popular sub-word tokenization algorithms: Byte Pair Encoding (BPE) algorithm (Sennrich et al., August 2016) and WordPiece algorithm (Wu et al., September 2016).

Sennrich et al. (August 2016) proposed Byte Pair Encoding (BPE) algorithm, which was originally proposed as a data compression algorithm (Gage, February 1994), to mitigate the OOV issue in machine translation. Given a fixed final token vocabulary size and a corpus, the BPE algorithm first splits all words into their characters and initializes the vocabulary with these characters (tokens). Next, it finds the most frequent co-occurrence of the tokens, merges those tokens, and adds the newly formed token to the vocabulary. This increases the vocabulary size by one. The algorithm repeats this procedure until it reaches the desired size limit of the vocabulary.

The WordPiece algorithm (Wu et al., September 2016) was proposed to solve the segmentation problem of the Korean and Japanese languages. This algorithm is similar to the BPE algorithm with the only the difference that WordPiece algorithm merges tokens together that could increase the likelihood of a unigram language model instead of the most frequent token bigrams. In Section 2.10.2 we will expand on the BERT model which is based on WordPiece and it is used in our experiments in Chapter 4.

2.9 Word Embeddings

We, humans, use natural language to communicate information. The textual representation of the language (characters, words, etc.) is comprehensible for us, however this is not the case for machines and learning algorithms as their primary means of communication is numerical.

One of the simplest ways to translate textual data to numerical values is one-hot encoding. For a vocabulary of size V, sorted in a given order, the one hot encoding of the i^{th} word is a V dimensional vector with a value of one in its i^{th} dimension and zeros everywhere else.

Although simple, the one-hot encoding representation is not a scalable solution. As the vocabulary size increases, the one-hot encoding representation requires much more memory and computational resources (Bengio et al., February 2003) and the representation does not reflect syntactic or semantic characteristics of the words. Embeddings (character (Kim et al., February 2016),

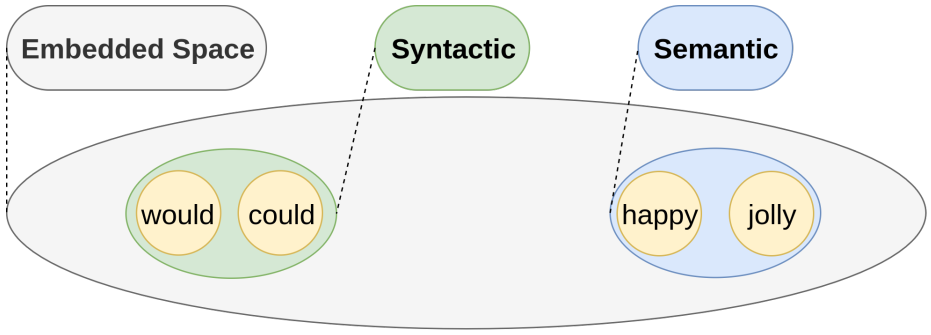


Figure 2.14: Word Embeddings are designed to capture semantic and syntactic relation between words.

word (Mikolov et al., May 2013), and sentence (Kiros et al., December 2015), etc.) were introduced to mitigate these problems by representing textual data via densely populated vectors and reducing the dimensionality of the representation vectors.

Word embeddings are dense vector representation of words. They are designed to capture syntactic and semantic similarities between words. Hence, as shown in Figure 2.14 similar words occupy the same region of the embedding vector space. There are two major word embedding model families in the literature: **statistical based embeddings** (Bullinaria and Levy, August 2007) and **context based embeddings** (Bengio et al., February 2003; Mikolov et al., May 2013).

Statistical based approaches leverage global statistical information of the corpus in order to form co-occurrence matrices with the assumption that words in the same contexts share similar semantics. For example in Latent Semantic Analysis (LSA) (Deerwester et al., March 1990) the matrices are of *word-document* type and in Hyperspace Analogue to Language (HAL) (Lund and Burgess, June 1996) the matrices are of *word-word* type. These large matrices are then decomposed to low-dimensional word representations through low-rank approximations.

Context based approaches (Bengio et al., February 2003; Mikolov et al., May 2013) learn word representations through a predictive neural model. The model is trained to predict a target word given a window of its local context (i.e. c words that appeared before and after the target word). The learned hidden parameters of this model constitute dense vector embeddings of the words in the vocabulary. The next sections will review work in context-based approaches as these have been used in our work (see Chapter 3).

Table 2.4: Examples of Target-Context pairs.

Target	**Context**
all	good, things
good	all, things, come
things	all, good, come, to
come	good, things, to, an
to	things, come, an, end
an	come, to, end
end	to, an

2.9.1 Word2Vec

Mikolov et al. (May 2013) proposed a shallow, two layer neural network to learn dense embedding representations for words. This context based approach to word embeddings aims to map words that share common contexts in the corpus to locations in the vector space that are close to one another. The training is performed using either of these two model architectures: **continuous bag-of-words** (CBOW) or **skip-gram**. Due to the low computational complexity of these shallow architectures, Word2Vec models can be trained on a large corpus in a short time (billions of words in hours) (Mikolov et al., May 2013).

Both the CBOW and the skip-gram models are given a fixed sized window of $(2 \times c + 1)$ words in a sentence. The word in the middle is called the *target* word and the words around it (i.e the c words before and after the target word) are called the context words. For example, Table 2.4 lists all the *target-context* pairs for a context window of size 2, for the sentence:

(7) All good things come to an end.

The CBOW model training objective is to predict the target word given the context around it. Assume we have a vocabulary of size V and we aim to learn unique dense word embeddings of length N for this vocabulary. Figure 2.15 shows the architecture of the CBOW model. In this model, the one hot representations of context words (of length V) constitute the input and the model output is the one-hot encoding of the target word.

The inputs will then be multiplied by a matrix of size $V \times N$, the embedding matrix, where each row corresponds to the embedding of a word in the vocabulary. The output of this multiplication is then passed to a hidden layer of size N followed by another matrix multiplication in order to produce

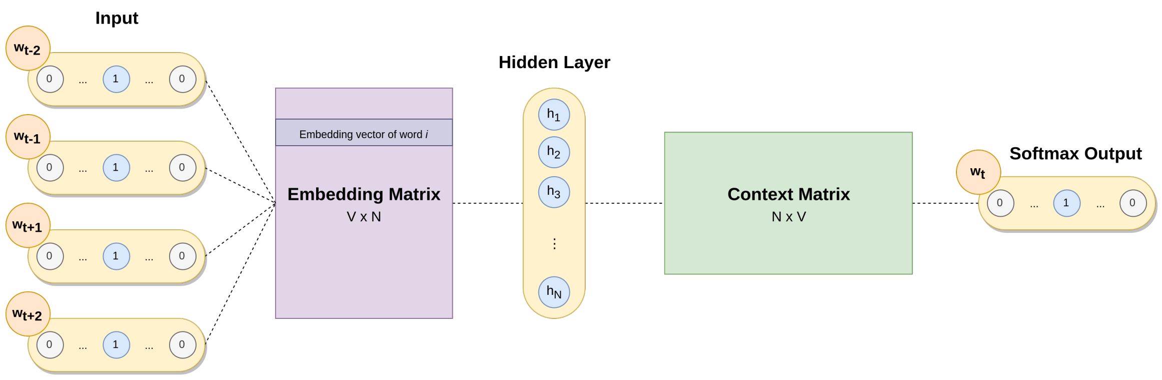


Figure 2.15: Architecture of The Word2Vec Continuous Bag-of-Words Model

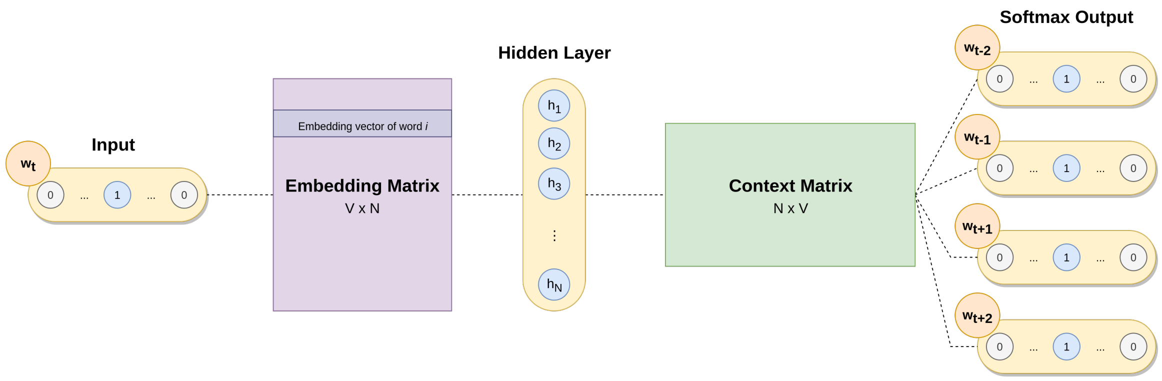


Figure 2.16: Architecture of The Word2Vec Skip-Gram Model

a probability distribution over the vocabulary for the most likely target word. CBOW models are fast to train and are suitable for large datasets (Mikolov et al., September 2013).

The training objective of the skip-gram model is to predict the context words of a given target word. Supposed as before, that we have a vocabulary of size V and we aim to learn unique dense word embeddings of length N for them. Figure 2.16 shows the architecture of the skip-gram model. In this model, the one-hot encoding of the target word constitutes the input and the output of the model is the context words. The model outputs a probability distribution over the entire vocabulary for each of the context words. For example, in the first row of Table 2.4, the target word *all* and the context words *good* and *things*, result in two training samples (i.e. `[all,good]` and `[all,thing]`) for which the model would need to produce two probability distributions over the vocabulary with highest values on the words *good* and *things*. The skip-gram model produces better quality embeddings when the training corpus is small (Moen and Ananiadou, December 2013).

2.9.2 GloVe

Pennington et al. (October 2014) proposed the Global Vector (GloVe) model in order to combine the statistical-based matrix factorization and the context-based skip-gram model together. The model aims to directly capture the global statistics of the corpus by modeling word embedding vectors through the ratio of word-word co-occurrence probabilities rather than the probabilities themselves. Similarly to the Word2Vec (see Section 2.9.1), the GloVe model captures the semantic relations of the words, but unlike Word2Vec, GloVe models these semantic relations based on the global co-occurrence of the words.

2.10 Generalized Language Models

In Section 2.9 we discussed two approaches to create word embeddings. The main shortcoming of these approaches is their inability to represent words in context. For example, in Sentences 8 and 9, the word *duck* has two different syntactic roles and meanings. In Sentence 8, the word *duck* is a noun and refers to a bird; on the other hand, in Sentence 9, the word *duck* is a verb and refers to the act of lowering head or one's body quickly. The approaches to word embeddings discussed in Section 2.9 would create the same vector for both instances of the word *duck*. In this section, we will discuss two approaches that were proposed to turn embeddings to a dynamic function of the context and make them more efficient in downstream NLP tasks.

(8) I fed the duck.

(9) I saw you duck a punch.

2.10.1 ELMo

Peters et al. (June 2018) proposed a multi-layer bidirectional LSTM based model that learns contextualized word embedding representations by pre-training a language model on a large corpus of data. The pre-training phase is an unsupervised training during which the model learns to predict the probability of next token given the past and future tokens. The pre-training phase can be scaled up since its learning objective is an unsupervised task and the unlabeled corpora can easily be expanded.

The hidden layers of the ELMo model constitute the word embeddings, which leads to word representations that are functions of the entire input sentence. For each specific downstream task, a model learns a linear combination of the hidden states of the ELMo model i.e. the learned weights of the linear combination of the ELMo layers indicate the needed task-specific modifications to the ELMo embeddings.

Peters et al. (June 2018) investigated the nature of the linguistic structures captured via ELMo embeddings by applying the model to semantic intensive and syntax intensive tasks. The ELMo embeddings were applied to word sense disambiguation (WSD) and part-of-speech (POS) tagging. The WSD experiments showed that the top layers of the model better capture semantic information of the language; on the other hand, the POS tagging experiments indicated that the lower layers of the model better represent syntactic information of the language. Since different layers of the embedding model represent different types of linguistic information and each downstream task has different linguistic needs, all layers of the ELMo model are always present in a new task. However, a task-customized model would need to learn a linear combination of these layers in order to optimize performance.

ELMo embeddings improve the performance of supervised learning tasks with small datasets. However, this embedding model relies on task-customized model architectures to learn and optimize the weights of the linear combination of the embedding hidden layers. This means that for every downstream task, a significant effort needs to be spent on searching for a good model architecture. In order to mitigate this issue, other language models such as ULMFiT (Howard and Ruder, July 2018), GPT (Radford et al., June 2018), and BERT (Devlin et al., June 2019) introduced the concept of fine-tuning, which we will discuss further in the next section.

2.10.2 BERT

In Section 2.10.1 we discussed the motivation behind the ELMo embedding model. This model transfers the contextualized embeddings to downstream tasks through customized task-specific neural architectures. This means that for every downstream task, significant effort is required to search

Table 2.5: Details of BERT model size variation.

Model	Transformer Layers	Self-attention Heads	Hidden Size	Total Parameters
BERT-Base	12	12	768	110M
BERT-Large	24	16	1024	340M

for a good model architecture. In order to mitigate this problem, Howard and Ruder (July 2018) proposed the ULMFiT model and explored the idea of using a pre-trained language model coupled with fine-tuning the same base model for all end tasks.

Following the same philosophy of ULMFiT, the GPT (Radford et al., June 2018), and later the BERT (Devlin et al., June 2019) models were proposed to eliminate the search for task-specific model architectures and instead use the pre-trained language model directly for all end tasks. These models follow a two step mechanism: pre-training and fine-tuning. In the pre-training step, the model is trained in an unsupervised fashion on a large corpus of data targeting language modeling tasks. In the fine-tuning step, the language model is augmented with a small neural structure and trained on task-specific data. In this section we will explain BERT as it was used in our work (see Chapter 4).

Devlin et al. (June 2019) proposed BERT, short for Bidirectional Encoder Representations from Transformers, a language model based entirely on the Transformer (see Section 2.7) architecture. The BERT architecture is comprised of multi-layer bidirectional Transformer encoder (the left side of Figure 2.13). BERT was introduced through two model sizes with the same architecture: **BERT-Base** and **BERT-Large**. Table 2.5 shows the main differences between these two variations.

Input Representation

As shown in Figure 2.17, the input to the model is represented as the sum of three embeddings: token embeddings, segmentation embeddings, and position embeddings.

Token embeddings are the WordPiece tokenization embeddings (see Section 2.8) which allows words to have variable length representations based on their morphemes and phonemes. In this approach words are seen as the sum of smaller sub-word units, allowing the model to better handle rare or unknown words.

[10]The image was taken from (Devlin et al., June 2019).

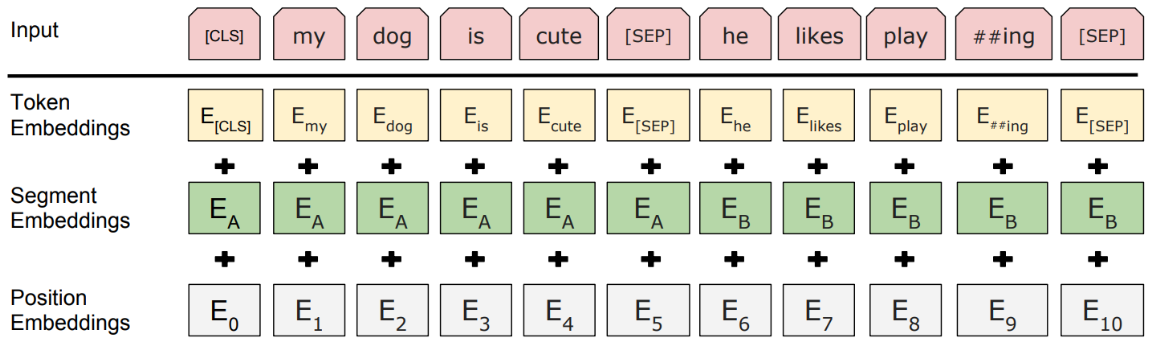


Figure 2.17: BERT input embeddings consisting of token embeddings, segmentation embeddings and position embeddings.[10]

Segmentation embeddings are motivated by the fact that many downstream tasks (e.g. Question Answering (QA), or Natural Language Inference (NLI)) are interested in the relation between two sequences, say sequence A and sequence B. The segmentation embedding forms an embedding for sequence A and sequence B separated by the special token `[SEP]`. If the downstream task contains only one sequence at each input, sequence A representation would correspond to the input, and sequence B would be $\emptyset$.

The BERT model essentially applies multiple Transformer blocks over the input sequences. In Section 2.7, we mentioned that the Transformer architecture does not contain any recurrence or convolution, hence this architecture is unaware of the order of the sequence. In order to augment the model with this information a positional embedding is added to the input embeddings.

The special token `[CLS]` is added at the beginning of each input in order to be used later for prediction in downstream tasks. The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks.

Pre-Training Phase

During the pre-training step, BERT is trained on two language model tasks: mask language model (MLM) and next sentence prediction (NSP).

MLM is designed to capture the word representations based on the context around a word (i.e. before and after). In this task, some tokens are masked and the model is trained to predict the masked tokens. Devlin et al. (June 2019) masked 15% of the tokens of each sequence, however replacing a token with special token [MASK] results in a mismatch between the pre-training phase and fine tuning phase since [MASK] does not appear during the fine-tuning step. In order to mitigate this issue, BERT uses the following heuristics during masking:

(1) Replace the chosen token with [MASK] token with the probability of 80%.

(2) Replace the chosen token with a random token with the probability of 10%.

(3) Keep the chosen token as is with the probability of 10%.

The output of the model is a sequence of tokens that the model predicts to be the masked tokens (i.e. the model does not reconstruct the entire input sequence), hence the output length is 15% of the input length.

NSP is motivated by the fact that many downstream tasks (e.g. QA, or NLI) are based on understanding the relationship between two sentences. In order to capture this relationship, BERT is pre-trained to predict whether one sentence is the next sentence of the other. For a sample sentence pair (A, B), half of the time B follows A and half of the time it does not. The output of the model for this task is a binary label indicating whether B follows A or not.

The training loss of the model is the sum of the mean MLM likelihood and the mean NSP likelihood. The pre-training phase can be scaled up, since its learning objectives are two unsupervised tasks and the the training data for both of these tasks can be trivially generated from any monolingual corpus.

Fine-Tuning Phase

The fine-tuning phase is seamless with only a few new parameters added to the pre-trained model. Figure 2.18 shows how the augmented pre-trained model converts the Transformer hidden states to the desired output of a downstream task.

For single sentence classification or sentence pair classification tasks (Figure 2.18.a and b), the final hidden representation of the special first token [CLS], $h_{[CLS]}$, is linearly transformed (i.e. multiplied by a weight matrix W) and passed to a Softmax function to predict the probability output classes ($\mathrm{Softmax}(h_{[CLS]}.W)$).

For the QA task (Figure 2.18.c), two probability distributions are defined over each tokens in the given paragraph of a given question in order to define the boundaries of the answer text span. One probability distribution estimates the probability of a token being the start of the text span and the other models the probability of a token being the end of the text span. The final hidden representation of the tokens of the paragraph will be multiplied by a weight matrix and passed through a Softmax function to determine the start and end tokens.

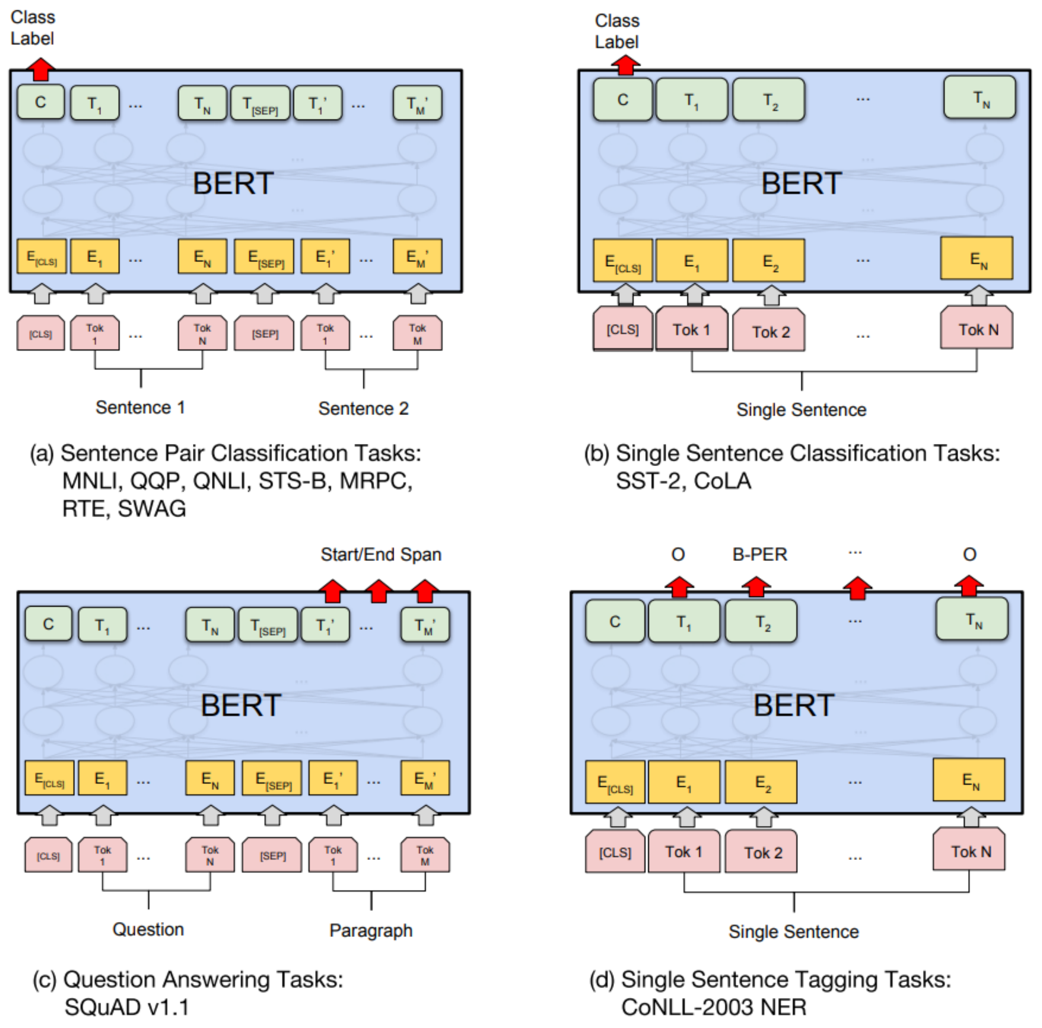
For token classification tasks (Figure 2.18.d) such as NER (see Chapter 4), a similar augmentation as the one for QA is performed. In token classification tasks, each output token belongs to either of k available classes, hence we need to define a probability distribution over the possible classes for each token in the output. The final hidden representation of the tokens will be multiplied by a weight matrix and Softmax function would determine the class of each token.

2.11 Neural Network Compression

Transferring knowledge from a pre-trained neural model, such as BERT (see Section 2.10.2) for NLP and VGG (Simonyan and Zisserman, September 2014) for computer vision, to a wide variety of downstream tasks is an effective method for improving the performance of learning algorithms. These models are pre-trained on large amounts of data, over tasks designed to capture the general structure of their target domain. They are then fine tuned on a small amount of downstream data, in order to bring their focus on a specific task. For example, as shown in Figure 2.19, a pre-trained BERT model can be fine tuned with an added layer to produce sate-of-the-art results for Named Entity Recognition (NER), Neural Machine Translation (NMT), or Question Answering (QA). (Devlin et al., June 2019).

The significant performance enhancement of fine tuning based models come at a cost of memory and computation complexity. Intuitively, each downstream task only needs a subset of the information which is learned and offered by the pre-trained model. Identification of the necessary information needed by a downstream tasks leads to smaller models and more efficient learning algorithms since we can discard the excess and unwanted information.

[11]The image was taken from (Devlin et al., June 2019).



Class Label
BERT
Sentence 1
Sentence 2
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
Class Label
BERT
Single Sentence
(b) Single Sentence Classification Tasks: SST-2, CoLA
Start/End Span
BERT
Question
Paragraph
(c) Question Answering Tasks: SQuAD v1.1
O
B-PER
O
BERT
Single Sentence
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 2.18: Fine-tuning BERT in downstream tasks with few new training parameters added to the base model and slightly modified training objective.[11]

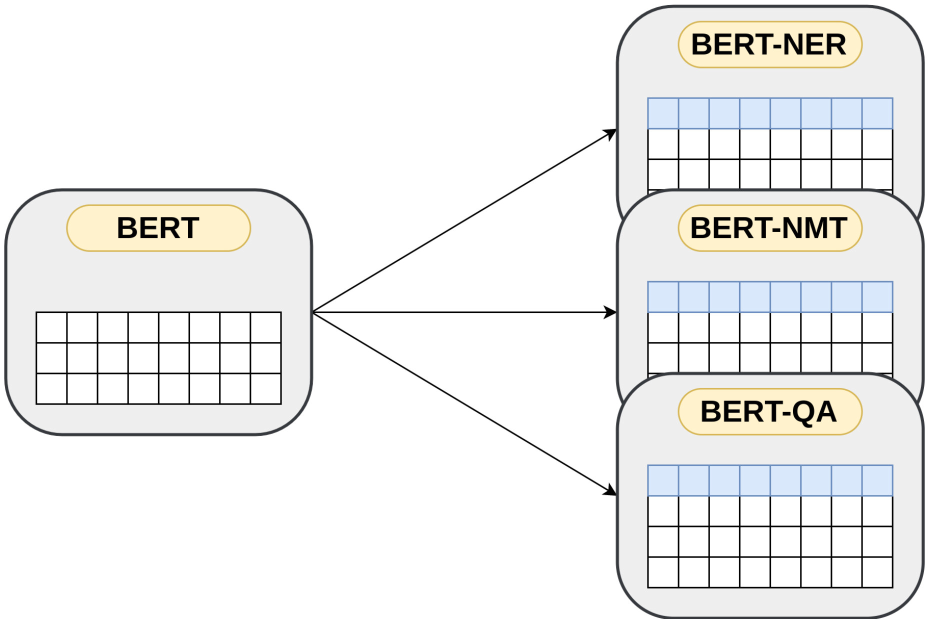


Figure 2.19: Pre-trained BERT model can be fine tuned with an added layer to produce sate-of-the-art results for Named Entity Recognition (NER), Neural Machine Translation (NMT), or Question Answering (QA).

Pruning unnecessary structures from the neural networks is a technique to reduce the memory and computational costs of the neural models without harm to performance (LeCun et al., November 1990; Han et al., December 2015). Pruning approaches can be categorized in two categories: structural pruning and parameter pruning.

Structural pruning modifies the structure of the network in order to reach smaller sub-networks while preserving performance. Structural pruning questions the importance of each neuron in the network and trims the less important neurons based on a criterion or a heuristic in order to achieve a lighter network (Molchanov et al., June 2019).

Parameter pruning reduces the number of weights or parameters of the network by removing the less important links in the network (Hassibi and Stork, December 1993; Li et al., April 2017). This technique preserves the general topology of the network, while producing a sub-network with sparse weight matrices. In this thesis we focus on parameter pruning (see Chapter 4).

Once the unwanted structures are removed from the model, the sub-network is retrained to adapt to the changes. However, training these sub-models from scratch (i.e. randomly initializing the weights that survived pruning) is not effective and usually leads to a lower performance than the original network (Li et al., April 2017). Initializing the connections of the network that survived pruning with the values of the initial training phase typically leads to better performance (Han et al., December 2015). This could be seen as a form of pre-training and fine tuning, in which the initialization schema of the network is equivalent to pre-training phase, and the fine tuning is the pruning task.

Frankle and Carbin (May 2019) showed that only a subset of the network parameters have impact on the model performance and thus the rest can be discarded. They formulated their findings as the *Lottery Ticket Hypothesis*, stating that a randomly initialized dense networks contains several sparse sub-networks, with varying performances, among which only a subset, the *winning tickets* can outperform all others when trained in isolation. Frankle and Carbin (May 2019) proposed the following iterative magnitude pruning (IMP) algorithm for finding the wining tickets:

(1) Randomly initialize a neural network with initialization values θ_0.

(2) Train the network for j iterations to reach an optimal performance with parameter configuration θ_j.

(3) Prune $p\%$ of the parameters (the ones with negligible weights) creating a mask m.

(4) The winning ticket initialization value is given by $m \odot \theta_0$.

The pruning algorithm proposed by Frankle and Carbin (May 2019) was able to recover the wining tickets on relatively small networks targeting vision tasks. However, they showed that the method is not able to find the winning tickets for larger networks without certain manipulations of the learning rate.

Frankle et al. (June 2019) augmented the IMP algorithm with a rewinding mechanism (IMPR) to mitigate the scalability and stability of IMP. In their approach the winning ticket initialization values are not necessarily a subset of θ_0 (see Step 4), but rather a subset of some θ_k, where k is some number of training iterations.

Their results underline the importance of the winning ticket initialization in highly parametrized networks. Their results suggest that the winning ticket initialization can be found early on during the training (0.1% to 7% through) and further training of the network does not have any significant impact on these initial values.

Frankle and Carbin (May 2019) and Frankle et al. (June 2019) conducted their search for winning tickets while training networks from scratch. In the context of transfer learning, where a network inherits most or all of its parameters from a well trained network, the concept of rewinding to an early stage of training for scalability and stability of IMP becomes irrelevant as the initialization are well trained in this context. In our work (see Chapter 4) we investigate the application of IMP in the context of transfer learning and therefore we will not use the rewinding mechanism in our experiments.

Frankle and Carbin (May 2019) and Frankle et al. (June 2019) have underlined the importance of the Lottery Ticket Hypothesis in achieving high performance light neural networks, however, their studies were focused on neural networks targeting vision tasks. To the best of our knowledge, there has not been any investigation on the performance of this conjecture on NLP oriented tasks. In Chapter 4, we will study the effect of Lottery Ticket Hypothesis on a highly parameterized network (BERT) targeting an NLP task (toponym detection) and propose a novel approach to insure stability and scalability of the IMP proposed by Frankle and Carbin (May 2019).

Chapter 3

Neural Network With Linguistic Insights

Our first attempt at toponym detection is based on the work of (Magge et al., July 2018), which constitutes the baseline system at SemEval 2019 shared task 12 (Weissenbacher et al., June 2019). We developed a Deep Feed-forward Neural Network (DFFNN) that uses domain-specific information as well as specific linguistic features. When evaluated with the SemEval 2019 shared task 12 test set (Weissenbacher et al., June 2019), its performance exceeded the baseline model's performance by 10.29% in F1 measure.

3.1 The DFFNN Model

The architecture of our first toponym recognition model is shown in Figure 3.1. The model pipeline is comprised of 2 main modules: an embedding layer, and a deep feed-forward network.

3.1.1 Embedding Layer

As shown in Figure 3.1, a word (e.g. *cartilage*) and its context (i.e. c words around it) constitute the input of the model. An embedding representation is constructed for each word in a document along with its context. More precisely, the embedding representation is comprised of a combination of word embeddings and feature embeddings.

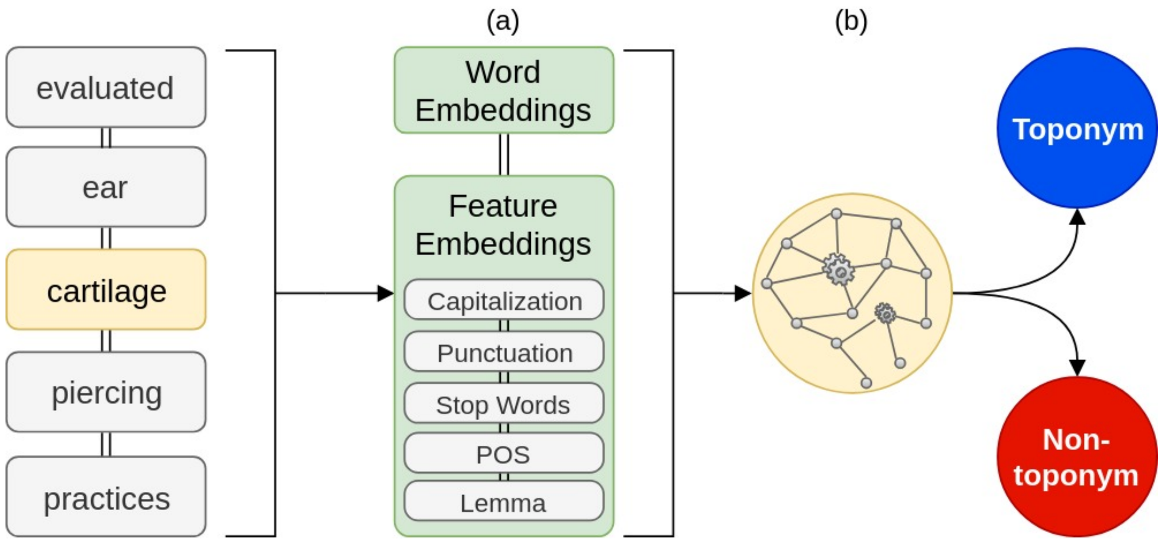


Figure 3.1: Our DFFNN toponym recognition model: A fixed context window of words is extracted, (a) an embedding is constructed for them, (b) then sent to a feed-forward neural network for predictions.

The model uses the pretrained Wikipedia-PubMed embeddings[1] for its word embedding component. Wikipedia-PubMed is trained on a corpus of $201,380$ words and it projects the representation vector on a 200 dimensional feature space. In order to capture more domain specific information (see Section 3.2) we used this embedding model as opposed to more generic embeddings such as Word2vec (Mikolov et al., May 2013) or GloVe (Pennington et al., October 2014) since the corpus used for training the Wikipedia-PubMed embedding consists partly of PubMed articles (Moen and Ananiadou, December 2013). This implies that the embeddings would be more appropriate when processing biomedical texts, and domain specific words. Moreover, the embedding model can better capture and represent the semantic association, closeness, and relation of words in biomedical articles. In order to form a single word embedding vector, the word embeddings of the target word and its context words are concatenated, resulting in a vector of size $200 \times (2c + 1)$, where c is the context size.

Previous works in toponym detection have shown how effective the use of particular linguistic features can be in this task (Magge et al., July 2018; Lieberman and Samet, December 2011). In order to leverage this information, our model is augmented with embedding for these features.

[1]`http://bio.nlplab.org/`

As shown in Figure 3.1, these include a capitalization feature, which captures whether a word is capitalized, uncapitalized, or written in uppercase letters (e.g. USA). Other linguistic features we observed to be useful (see Section 3.2) include punctuation, stop words, part of speech tags, and the word embedding of the lemma of the word. These linguistic features are encoded using a binary vector representation, for example in the case of capitalization feature, if a word is capitalized, its feature embedding is $[1, 0]$ otherwise it is $[0, 1]$ and if it is written in all uppercase letters, then its feature embedding is $[1, 1]$. The final embedding representation of the input is made of the concatenation of the word embedding vector and the feature embedding of the input word and its context words. As shown in Figure 3.1, this embedding representation, is passed to the next layer of the model, the DFFNN.

3.1.2 The Architecture

The DFFNN receives the concatenated embedding representation formed in the embedding layer (see Section 3.1.1) and performs a binary classification. This component contains 3 hidden layers and one output layer, where each hidden layer contains 500 ReLU activation nodes (see Table 2.1). Upon receiving an input vector x the output $h(x)$ of a hidden layer h is computed as:

$$h(x) = \mathrm{ReLU}(Wx + b) \tag{4}$$

Recursive application of the above equation for all 3 hidden layers defines the DFFNN model. A 2 dimensional softmax activation function forms the output layer of the network, which produces the output $O(x)$ upon receiving the input x as follows:

$$O(x) = \mathrm{Softmax}(Wx + b) \tag{5}$$

The Softmax function was used at the output layer since this function provides a categorical probability distribution over the available classes for an input x, i.e.:

$$p(x = \text{toponym}) = 1 - p(x = \text{non-toponym}) \tag{6}$$

Table 3.1: Optimal hyper-parameters of DFFNN model.

Parameters	**Value**
Learning Rate	0.01
Batch Size	32
Optimizer	SGD
Momentum	0.1
Loss	Weighted Categorical cross-entropy
Loss weights	$(2, 1)$ for toponym vs. nontoponym

In order to prevent overfitting, we employed two mechanisms: drop-out and early-stopping. Early-stopping prevents over-fitting and poor generalization by stopping the training process if the loss on the development set (see Section 3.2) starts to rise. Drop-out aims to avoid the inner dependencies between neurons in the network, leading to a more robust and stable training, the probability of drop-out was set to 0.5 in our training. Norm clipping (Pascanu et al., June 2013), which scales the gradient when its norm exceeds a certain threshold, was also used to prevent the occurrence of exploding gradient. We experimentally found the best performing threshold for norm clipping to be 1 for our model.

In our experiments, we investigated the performance of various model architectures both in depth and number of hidden units per layer as well as other hyper-parameters listed in Table 3.1. We observed that deepening the model leads to immediate over-fitting due to the small size of the dataset (Hinton et al., July 2012) (see Section 3.2) even with the presence of a drop-out function to prevent it. Table 3.1 presents the optimal hyper-parameter configuration with the development set used to fine tune them.

3.2 Experiments and Results

For comparative purposes, as indicated in Section 2.4, a baseline model for toponym detection was provided by the organizers of SemEval 2019 shared task 12 (Weissenbacher et al., June 2019). The baseline, inspired by (Magge et al., July 2018), also uses a DFFNN architecture comprised of only 2 hidden layers and 150 ReLU activation functions per layer.

Table 3.2 shows the performance of our basic model presented in Section 3.1.2 (see row #4) compared to the official SemEval baseline (row #3). A series of experiments was carried out

Table 3.2: Performance score of the baseline, our proposed model and its variations. The suffixes represent the presence of a feature, P: Punctuation marks, S: Stop words, C: Capitalization features, POS: Part of speech tags, W: Weighted loss, L: Lemmatization feature. For example DFFNN Basic+P+S+C+POS refers to the model that only takes advantage of punctuation marks, stop words, capitalization feature, and part of speech tags.

#	Model	Context	Precision	Recall	F1
8	DFFNN Basic+P+S+C+POS+L+W	5	80.69%	79.57%	80.13%
7	DFFNN Basic+P+S+C+POS+L	5	77.57%	73.44%	75.45%
6	DFFNN Basic+P+S+C+POS	5	77.55%	70.37%	73.79%
5	DFFNN Basic+P+S+C	2	78.82%	66.69%	72.24%
4	DFFNN Basic+P+S	2	79.01%	63.25%	70.26%
3	SemEval Baseline	2	73.86%	66.24%	69.84%
2	DFFNN Basic+P	2	74.70%	63.57%	68.67%
1	DFFNN Basic+S	2	64.58%	64.47%	64.53%

to evaluate the influence of a variety of parameters on the performance of the model, which are described in the next sections.

3.2.1 Effect of Domain Specific Embeddings

As indicated by much work in this field of research (e.g. (Amitay et al., July 2004; Purves et al., June 2007; Qin et al., November 2010; Kienreich et al., July 2006; Garbin and Mani, October 2005)), the task of toponym detection is highly dependent on the discourse domain. Due to this, our basic model employs the Wikipedia-PubMed embeddings. In order to measure the effect of such domain specific information, we experimented with 2 other pretrained word embedding models: Google News Word2vec (Google, 2019), and a GloVe model trained on Common Crawl (Pennington et al., 2014). As shown in Figure 3.2, although, the Wikipedia-PubMed has a smaller vocabulary in comparison to the other embedding models, with the SemEval dataset, it suffers significantly less from out of vocabulary words (OOV) since it was trained on a closer domain.

We experimented with our DFFNN model using each of these embeddings and optimized the context window size to achieve the highest F-measure on the development set. Figure 3.3 shows the performance of these models on the test set. As expected, the Wikipedia-PubMed performs better than the other embedding models. This is likely due to its small number of OOV words (see Figure 3.2) and its domain-specific knowledge. As Figure 3.3 shows, the performance of the GloVe model is quite close to the performance of Wikipedia-PubMed. In order to investigate this further,

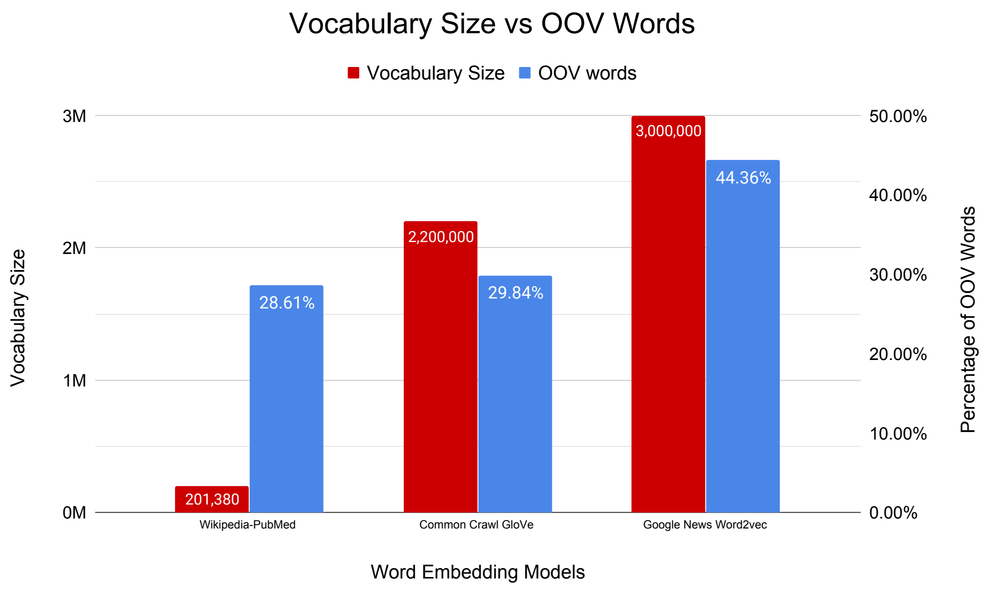


Figure 3.2: A comparison of the vocabulary size of word embedding models and their percentage of OOV words with respect to the dataset.

we trained another model on the combination of the two embeddings. As shown in Figure 3.3, the performance of this combined model (Wikipedia-PubMed + GloVe) is higher than the GloVe model alone but lower than the Wikipedia-PubMed. The observed decrease in performance can be attributed to the dilution of the domain specific information captured by the Wikipedia-PubMed embeddings, since GloVe embeddings provide relatively more general representations, hence when the network is presented with a combination of the two embeddings, it will try to adapt a more unified predictive strategy, leading to loss in performance. Because of this, from here on, our experiments were carried exclusively using Wikipedia-PubMed word embeddings.

3.2.2 Effect of Linguistic Features

Although deep learning approaches have lead to significant improvements in many NLP tasks, simple linguistic features are often very useful. In the case of NER, punctuation marks constitute strong signals (Gelernter and Balaji, January 2013). In order to investigate the effect of punctuation marks on our model, we retrained the DFFNN Basic without punctuation information. As Table 3.2 (row #1) shows, the removal of punctuation marks, decreased the F-measure from 70.26%

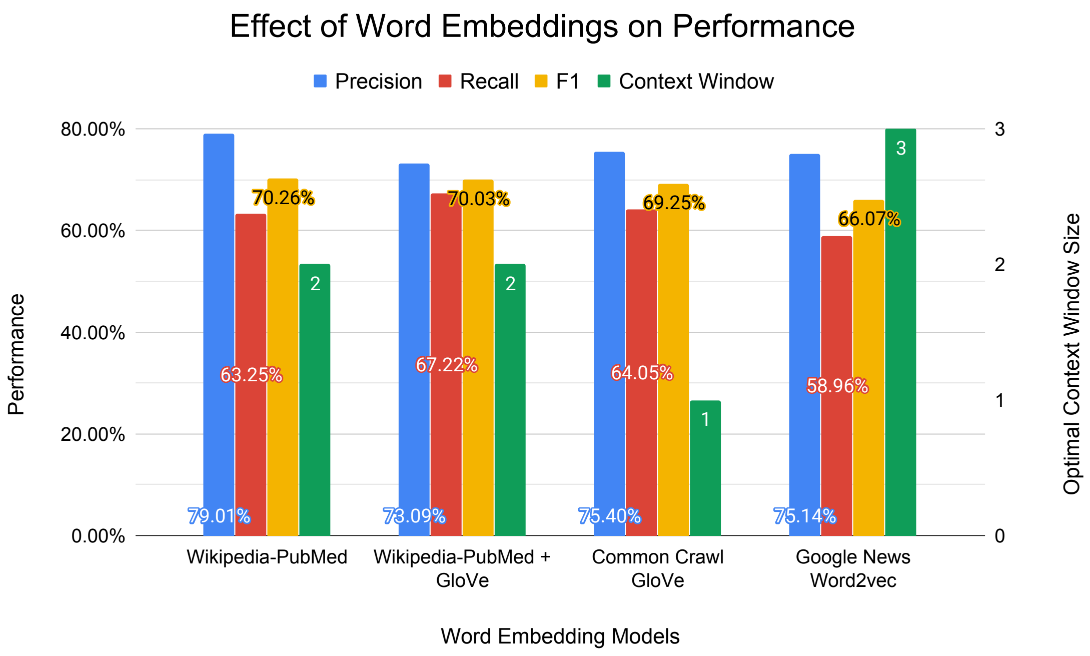


Figure 3.3: Effect of word embeddings on the performance of our proposed model architecture.

to 64.53%. A manual error analysis showed that many toponyms appear inside parenthesis, near a dot at the end of a sentence, or after a comma. Hence, as suggested in (Gelernter and Balaji, January 2013) punctuation is a good indicator of toponyms and should not be ignored.

As Table 3.2 (row #2) shows, the removal of stop words, does benefit the performance of the model. In fact it leads to a notable decrease in F-measure performance of the model (from 70.26% to 68.67%). We hypothesize that some stop words such as *in* do help the system detect toponyms as they provide a learnable structure for detection of toponyms and that is why the model accuracy suffered once the stop words were removed.

As seen in Table 3.2 (row #4) although our basic model has a high precision of 79.01%, it does suffer from a low recall of 63.25%. A manual inspection of the toponyms in the dataset revealed that either their first letter is capitalized (e.g. *Canada*) or they are written in all uppercase letters (e.g. *USA*). In an attempt to help the DFFNN learn more structure from the small dataset, we provided the model with this information (see Section 3.1.1). As a result the recall increased from 63.25% to 66.69% and the F1 performance increased from 70.26% to 72.27% (see Table 3.2 #5).

We experimented with the use of part of speech (POS) tags as part of our feature embeddings to assist the neural network better understand and model the structure of sentences. We used the NLTK POS tagger (Bird et al., June 2009) which uses the Penn Treebank tagset. As indicated in Table 3.2 (row #6), the POS tags significantly improve the recall of the network (from 66.69% to 70.37%) hence leading to a higher performance in F1 (from 72.24% to 73.79%). We observed that the presence of the POS tags help the DFFNN to better learn the structure of the sentences and take advantage of more contextual information (see Section 3.2.3) in its predictions.

Neural networks require large datasets to learn structures and they learn better if the dataset contains similar examples so that the system can cluster them in its learning process. Since our dataset is small and the Wikipedia-PubMed embeddings suffer from 28.61% OOV words (see Table 3.2), we tried to help the network better cluster the data by adding the lemmatized word embeddings of the words to the feature embeddings and see how our best model reacts to it. As shown in Table 3.2 (row #7), this improved the F1 measure significantly (from 73.79% to 75.45%).

3.2.3 Effect of Window Size

We experimented with the amount of contextual information presented to the network by varying the size of the context window. As seen in Figure 3.4, the best performance of our Basic Model is achieved with $c = 2$. Providing the model with more contextual information (i.e. considering $c > 2$) leads to overfitting as the model is unable to extract any meaningful structures from the extra information. The small size of the data set does not allow the DFFNN to learn the structure and composition of the sentences, hence increasing the context window alone does not help the performance. However, a window size of $c = 2$ allows the model to focus on the local structure of the named entities. In order to help the neural network better understand and use the contextual structure in its predictions, we experimented with part of speech (POS) tags as part of our feature embeddings. As shown in Figure 3.4, the POS tags help the DFFNN to take advantage of more contextual information, as a result the DFFNN with POS embeddings achieves a higher performance on larger window sizes. The context window for which the DFFNN achieved its highest performance on the development set was $c = 5$, and on the test set the performance was increased from 72.24% to 77.10% (see Table 3.2 #6).

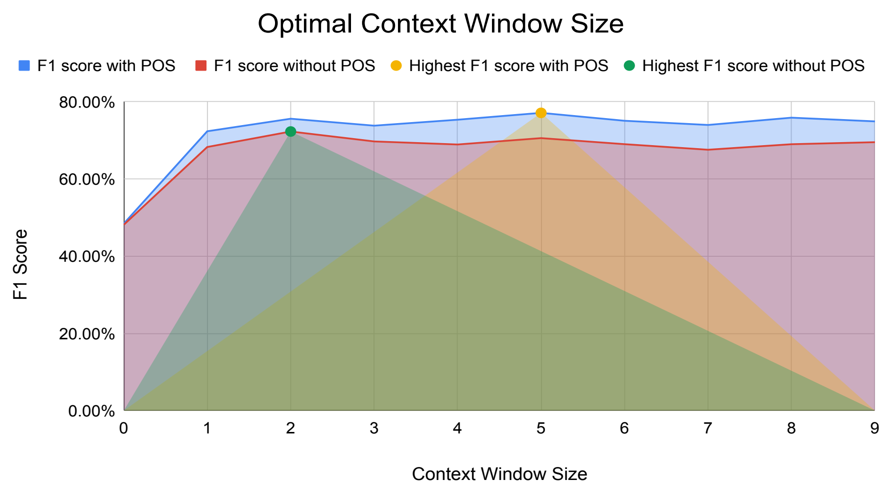


Figure 3.4: Effect of the context window on the performance of the DFFNN model with and without POS features. (DFFNN Basic+P+S and DFFNN Basic+P+S+C+P)

3.2.4 Effect of the Loss Function

As shown in Table 3.2 most models suffer from a lower recall than precision. The main cause of this gap in performance is the highly unbalanced distribution of samples, i.e. the number of non-toponym words are much higher than toponyms (99% vs 1%). Therefore, the neural network prefers to optimize its performance by concentrating its efforts on correctly predicting the labels for the dominant class (non-toponym). To address this, we experimented with a weighted loss function to minimize the gap between the recall and precision of the models. We adjusted the importance of predicting the correct labels experimentally and found that by weighing the toponyms 2 times more than the non-toponyms, the system reaches an equilibrium between the precision and recall measures, leading to a higher F1 performance. This is indicated by “W” in Table 3.2 row #8.

3.2.5 Overall Model Reliance on Linguistic Features

Section 3.2.2 demonstrated the influence of certain linguistic features on the performance of the model. Table 3.2 shows the influence of these features on performance as the model parameters are iteratively changed. However, it does not indicate the final contribution of the features to the model, since certain features can overlap on the hidden structures they represent. In this section, we measure the reliance of the final learned model on these linguistic features. To do this, we used the

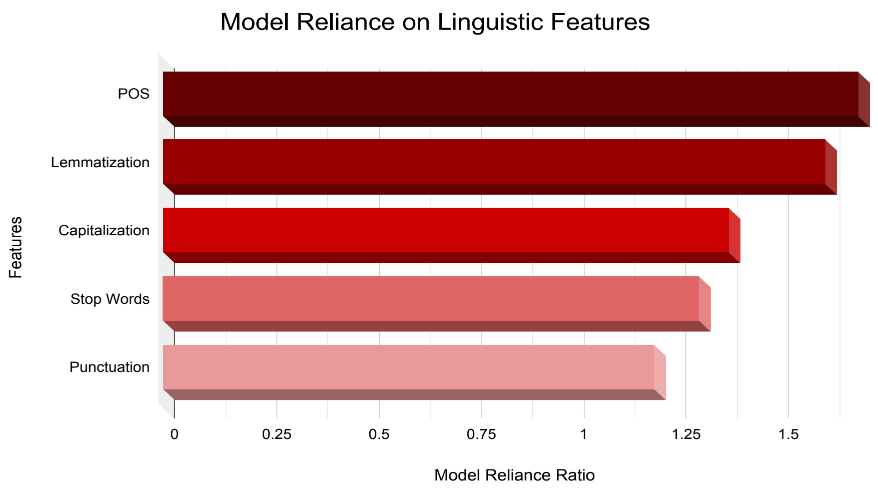


Figure 3.5: Model reliance of the DFFNN model on the linguistic features ranked in order of decreasing reliance.

unbiased statistical measure for model reliance (MR) introduced in (Fisher et al., October 2019). The MR of feature f is defined as:

$$MR(f) = \frac{e_{permuted}(f)}{e_{original}} \tag{7}$$

In this equation, $e_{original}$ refers to an error measure such as $(\text{error} = 1 - \text{F1}_{\text{score}})$ over the test set. The $e_{permuted}(f)$ is the error measure over the test set while the feature f is randomized by permutation over all samples, and other features are untouched. An MR ratio of 1 or close to 1 for a feature f indicates that the learned model does not rely on f to make its predictions. Figure 3.5 shows the MR of our proposed model. The MR value of the features was used to rank the importance of these features to the final prediction of the model. As Figure 3.5 shows, the most reliable features are grammatical in nature (POS and Lemma) whereas punctuation constituted the least reliable feature.

3.3 Discussion

Overall our best model (DFFNN #8 in Table 3.2) is composed of the basic DFFNN plus all linguistic features and a weighted loss function. The experiments and results described in Section 3.2

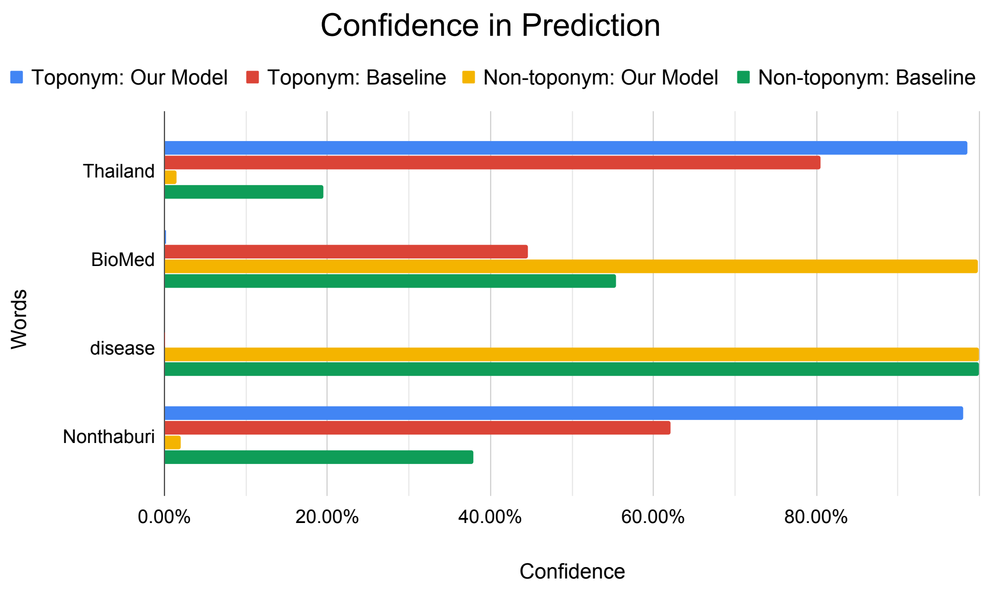


Figure 3.6: Confidence of the DFFNN model and the baseline model in their categorical prediction on four randomly selected words, Thailand (toponym), BioMed (non-toponym), disease (non-toponym), and Nonthaburi (toponym).

underline the importance of domain specific word embedding models. These models reduce OOV words and also present us with embeddings the capture the relation of the words and relevant knowledge they represent in the specific domain of study.

Our experiments also underline the importance of linguistic insights in the task of toponym detection. These insights and features should ideally be learned by the system itself, however, when the data is scarce, as it was in our case, we should take advantage of the hidden linguistic structures of the data for better performance. Moreover, in order to visualize and compare the confidence of our proposed model with the baseline model in their prediction as given by the softmax function (see Equation 5), we randomly picked 2 toponyms (*Thailand* and *Nonthaburi*) and 2 non-toponyms (*BioMed* and *disease*) and investigated the results. As Figure 3.6 shows, our model produces much sharper confidence in comparison to the baseline model.

3.4 Conclusion

This chapter presented our linguistic enhanced deep learning approach for toponym identification. The approach was evaluated using the dataset of the SemEval task 12 shared task on toponym resolution (Weissenbacher et al., June 2019). Our best DFFNN approach took advantage of domain specific embeddings as well as linguistic features and achieved a significant increase in F-measure compared to the shared-task baseline system (from 69.74% to 80.13%).

We chose the feed-forward architecture based on the hypothesis that localized contextual information is sufficient to identify the toponyms. In Chapter 4, our second attempt at toponym detection, we will experiment with neural architectures that considered all available contextual information (Vaswani et al., January 2017; Lan et al., September 2019) within a meaningful unit of language.

Chapter 4

Efficient Toponym Identifiers for Medical Domain Using BERT

In our second attempt at toponym detection, we used the BERT model (Devlin et al., June 2019) (see Section 2.10.2) as the backbone of our toponym identifier system. We created a family of **T**oponym **I**dentification **M**odels based on **BERT** (TIMBERT) as the core of the network, in order to learn directly in an end-to-end fashion the mapping from the input sentence to the associated output labels. When evaluated with the SemEval 2019 shared task 12 test set (Weissenbacher et al., June 2019), our best model achieves an F1 score of 90.85%, a significant improvement compared to our DFFNN (see Chapter 3) model's performance of 80.13% and the state-of-the-art 89.10% (Wang et al., June 2019; Magnusson and Dietz, June 2019).

The significant performance enhancement of our BERT based model comes at a cost of memory and computation complexity. Pruning unnecessary structure from neural networks has been a popular way to reduce the memory and computational costs of the neural models without harm to accuracy (see Section 2.11). In this chapter we propose a novel algorithm for parameter pruning and investigate its application on the toponym recognition task.

4.1 TIMBERT Model

In Chapter 3, we presented our DFFNN approach for toponym identification. This model relied on domain specific embeddings, linguistic insights, and localized contextual information. TIMBERT was developed in order to sidestep these brittle design choices and to learn directly in an end-to-end fashion the mapping from an input sentence to the associated output labels while maintaining a compact structure.

4.1.1 Embeddings

Previous attempts to toponym detection in medical domain have confirmed the benefits of using domain specific word embeddings, specifically Wikipedia-PubMed embeddings[1] (Davari et al., April 2019; Wang et al., June 2019; Magnusson and Dietz, June 2019). This embedding model closely captures the semantic association, closeness, and relation of words in medical articles. However, there are two limitations to this approach:

- This approach views the task of toponym detection as highly dependent on the textual domain. Hence, it reduces the generalizability of the model to other domains and requires domain specific word embeddings for each domain of interest.

- The pretrained embeddings suffer from a significant number of out of vocabulary words. In our case, (see Section 3.2.1), 28.61% of the corpus vocabulary were OOV words and were mapped to a single token (`[UNK]`), which provides the system with no information about the initial token.

In order to mitigate these problems, we used the pretrained BERT model (Devlin et al., June 2019) (see Section 2.10.2) as the backbone of our architecture. In the pretraining phase, BERT learns a general representation of the language. This general representation is then fine-tuned for a specific task and domain. Fine tuning allows BERT to generalize across different domains, while other approaches (eg. Word2Vec (Mikolov et al., May 2013), GloVe (Pennington et al., October 2014) (see Section 2.9)) fail to do so and require a single language model for each domain to achieve optimal performance.

[1] http://bio.nlplab.org/

In this chapter, we treat the problem of toponym detection as an open vocabulary problem, allowing words to have variable length representations based on their morphemes and phonemes. We use WordPiece embeddings (Wu et al., September 2016) (see Section 2.8) with a 30,000 token vocabulary. Representing morphologically complex words as the sum of their morphemes leads to a compact network vocabulary and improves cross domain generalizability of the model (Sennrich et al., August 2016). For example, a conventional word embedding model (Mikolov et al., May 2013) would learn a vector representations for the word *biology*, but WordPiece based embeddings (Wu et al., September 2016) would treat it as:

$$\text{Biology} = \text{Bio} + \text{logy}$$

The WordPiece approach, would learn embedding representations for the sub-words *bio* (life) and *logy* (denoting a subject of study or interest). It then represents the word *biology* as the concatenation of the learned sub-word embeddings.

4.1.2 Linguistic Features

As shown in Chapter 2, previous works on toponym detection in the medical domain have typically taken advantage of handcrafted features to achieve competitive performance (Magge et al., July 2018; Davari et al., April 2019; Wang et al., June 2019; Magnusson and Dietz, June 2019). BERT has demonstrated great capability to capture linguistic features (Clark et al., August 2019) and transferring the learned knowledge to downstream tasks. Aimed to develop an end-to-end toponym detection model with minimal reliance on feature engineering, we chose BERT as the backbone of our second model. We experimented with the influence of different linguistic features on the performance of the model, to verify whether the model needed additional linguistic features or not.

4.1.3 Contextual Information

In Chapter 3, the DFFNN model was based on the hypothesis that only localized contextual information is needed for the identification of the toponyms. Our experiments were based on a model that had only access to a sliding window of information. In this chapter we were motivated to experiment with neural architectures that considered all available contextual information (Vaswani et al., January 2017; Lan et al., September 2019) within a meaningful unit of language. In Section 2.7 we discussed the Transformer model and its ability to deal with variable length inputs, while being much more parallelizable in comparison to RNNs. Hence, we decided to base the architecture of TIMBERT on the Transformer model. As discussed in Section 2.10.2, the BERT architecture is based on the Transformer model, therefore we chose to have it as the backbone of our architecture.

4.1.4 Network Pruning

Pre-trained language representations such as GPT (Radford et al., June 2018), BERT (Devlin et al., June 2019), and XLNET (Yang et al., December 2019) have shown substantial performance improvements in a variety of NLP tasks. Knowledge transfer from these language representations to downstream tasks is seamless by adding a single task-specific output layer to the base structure. However, the significant performance enhancement of these pre-trained language representations comes at a cost of memory and computation complexity. For example, the BERT-Base model and BERT-Large model contain $110M$ and $340M$ parameters respectively.

Intuitively, each downstream task only needs a subset of the information which is learned and offered by the pre-trained language model. For example, some tasks may require more semantic information (e.g. word sense disambiguation) and some task may require more syntactic information (e.g. part-of-speech (POS)) (Peters et al., June 2018). Identification of the necessary information needed by a downstream task leads to smaller and more efficient learning algorithms since we can discard the excess and unwanted information.

Motivated by this intuition, we experimented with an iterative magnitude pruning (IMP) algorithm (see Section 2.11) to identify the unnecessary information and discard them. In our experiments, we augmented this algorithm for better stability and performance (see Section 4.3.2).

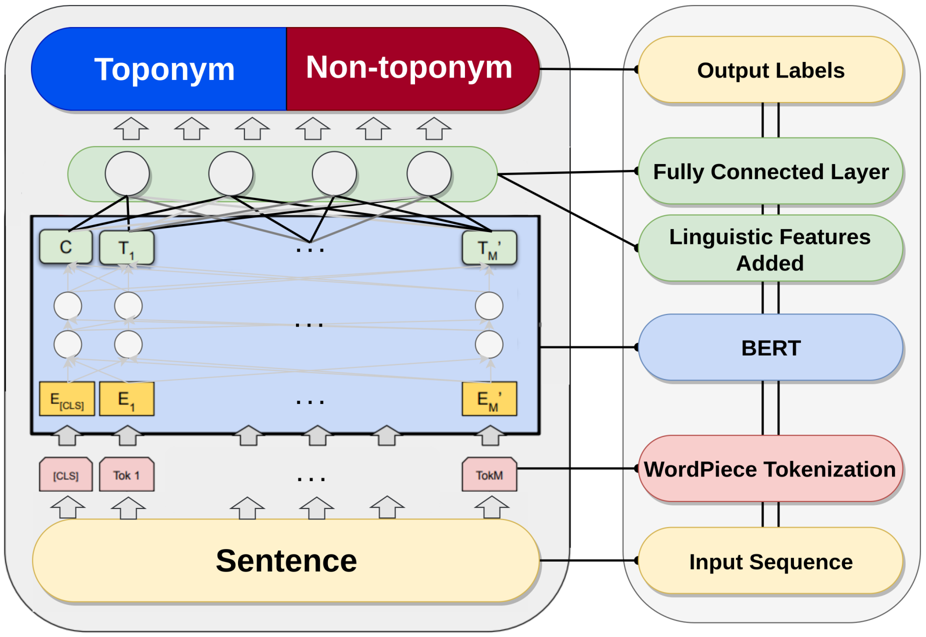


Figure 4.1: TIMBERT model architecture.

4.2 TIMBERT Architecture

The architecture of our toponym recognition model is shown in Figure 4.1. The WordPiece tokenization of a sentence constitutes the input of the model. These tokens are then passed to a pre-trained BERT network (see Section 2.10.2). The output of the network along with certain linguistic features are then passed to a fully connected layer which determines the labels of each token. We call our model TIMBERT: Toponym Identification Model based on BERT. In our experiments, we used two variations of the BERT model: BERT-Base, and BERT-Large (see Section 2.10.2). The respective TIMBERT models are called TIMBERT-Base and TIMBERT-Large. Since the BERT-Large model is much more computationally expensive than BERT-Base, we tried to limit the experiments involving this model.

4.3 Experiments and Results

For comparative purposes, we will use the DFFNN model developed in Chapter 3 as a baseline for toponym detection. Table 4.1 shows the performance of our basic model i.e. TIMBERT-Base without any linguistic features that was presented in Section 4.2 (see row #6) compared to the

Table 4.1: Performance of the baseline DFFNN model (see Chapter 3), and the TIMBERT based models.

#	Model	Precision	Recall	F1
1	TIMBERT-Large-CoNLL-w/-Orthographic-Pruned	90.51%	91.19%	90.85%
2	TIMBERT-Large-CoNLL-w/-Orthographic	89.73%	90.23%	89.98%
3	TIMBERT-Large-w/-Orthographic-Pruned	83.76%	86.89%	85.25%
4	TIMBERT-Large-w/-Orthographic	83.41%	86.88%	85.11%
5	TIMBERT-Base-w/-Orthographic	82.61%	83.19%	82.90%
6	TIMBERT-Base	82.59%	80.17%	81.36%
7	TIMBERT-Base-w/-POS	81.96%	80.08%	81.01%
8	DFFNN	80.69%	79.57%	80.13%
9	TIMBERT-Base-w/o-Punctuation	80.04%	78.75%	79.39%
10	TIMBERT-Base-w/o-Stop-Words	72.14%	72.01%	72.08%

baseline (see row #8). A series of experiments was carried out to evaluate the influence of a variety of parameters on the performance of the model, which are described in the next sections.

4.3.1 Effect of Linguistic Features

Previous works on the SemEval 2019 shared task 12 dataset (Weissenbacher et al., June 2019) have underlined the importance of carefully handcrafted linguistic features in order to achieve competitive performance (Magge et al., July 2018; Davari et al., April 2019; Wang et al., June 2019; Magnusson and Dietz, June 2019). In this section we examine the influence of selected linguistic features on the performance of TIMBERT-Base (see Table 4.1 row #6).

4.3.1.1 Orthographic Features

The orthographic features we experimented with target the capitalization of the letters within a token word. This feature is presented to the model as a one-hot vector and it captures whether a word is capitalized (e.g. Canada), uncapitalized (e.g. city), or written in uppercase letters (e.g. USA). Since in the preprocessing of the data, all tokens are brought to lowercase, the TIMBERT-Base is unaware of this feature. Our experiments showed that augmenting the model with the orthographic information results in the increase of the F1 performance from 81.36% to 82.90% (see Table 4.1 row #5).

4.3.1.2 Part of Speech Tags

Our experiments in Chapter 3 indicated that augmenting the model with part of speech (POS) tags results in an improvement in the performance of the model. On the other hand, the BERT model has shown a great ability to capture a number of linguistic features and transferring them to downstream tasks (Clark et al., August 2019). Since BERT constitutes the backbone of TIMBERT, we investigated whether or not our model is aware of POS tags.

We used the NLTK POS tagger (Bird et al., June 2009) which uses the Penn Treebank tagset (Marcus et al., June 1993). As indicated in Table 4.1 (row #7), including the POS tags reduced the performance of the model from the F1 of 81.36% to 81.01%. This suggests that the TIMBERT-Base model is already aware of the POS tags since the augmentation of the model with POS tags does not affect its performance to any statistical significance. Although statistically insignificant, the slight decrease in the performance could be due to the errors introduced by the NLTK POS tagger model (Bird et al., June 2009) and the disagreement between the domain that the tagger was trained on (Brown Corpus (Francis and Kucera, 1964), news paper articles) and the domain that it was tested on (medical journal articles).

4.3.1.3 Stop Words

As Table 4.1 (row #10) shows, the removal of stop words using the NLTK stop words corpus (Bird et al., June 2009) of 179 words, does not benefit the performance of the model. In fact it leads to a significant decrease in F1 performance of the model (from 81.36% to 72.08%). We hypothesize that some stop words such as *in* do help the system detect toponyms as they provide a learnable structure for detection of toponyms and that is why the model accuracy suffered once the stop words were removed. Moreover, the BERT model has evolved to capture contextual information and language structures. Therefore, to observe its true power, it should be given fully comprehensible and structured sentences. stop words removal distorts sentence structure and therefore harms the model performance.

4.3.1.4 Punctuation

In order to investigate the effect of punctuation marks on our model, in the preprocessing step we removed all punctuation marks. We then trained the TIMBER-Base without any punctuation information. As Table 4.1 (row #9) shows, the removal of punctuation marks, decreased the F1 from 81.36% to 79.39%. A manual error analysis showed that many toponyms appear inside parenthesis, near a dot at the end of a sentence, or after a comma (e.g. *(Montreal, Canada)*). Hence, as suggested in (Davari et al., April 2019; Gelernter and Balaji, January 2013) punctuation is a good indicator of toponyms and should not be ignored.

4.3.2 Effect of Network Pruning

Experiments on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., November 2018), which is a collection of diverse natural language understanding tasks, have shown great performance gains when the backbone model was switched from BERT-Base to BERT-Large (Devlin et al., June 2019). Motivated by these findings, we investigated the impact of BERT-Large for our task. We substituted the BERT-Base backbone module with BERT-Large in our best performing model from Section 4.3.1 (i.e. TIMBER-Base-w/-Orthographic). As indicated in Table 4.1 (row #4), the resulting model, TIMBER-Large-w/-Orthographic, improved the F1 performance from 82.90% to 85.11%.

The significant performance enhancement of this model came at a cost of memory and computation complexity. As indicated in Section 2.10.2, BERT-Large has 340M parameters which is 3 times more than BERT-Base. In an effort to find lighter models while preserving performance, we experimented with the iterative magnitude pruning (IMP) algorithm outlined by Frankle and Carbin (May 2019) (see Section 2.11).

4.3.2.1 Sensitivity of IMP towards the ordering of the training data

Recall from Section 2.11, that the iterative magnitude pruning (IMP) algorithm identifies and removes unnecessary model parameters by training the model and scanning for parameters with zero or close to zero weights. In our experiments, we did not use a fixed number of training iterations and instead used early stopping to end the training process. The weights of the trained model were then ordered and the top $p\%$ of the smallest weights were removed. However, we realized that the weight ordering is highly dependent on the training dataset order and the optimization method, i.e. shuffling the dataset and repeating the experiment resulted in different weight orderings.

In order to have a measure for this observation, we trained a total of eleven models, and sorted the trained weights. We kept the weight ordering of one trained model as the reference and computed our statistics using the weight ordering of the other ten trained models (repeated trials). Let W^{ref} and W^t be the sets of all trained parameters for the reference model and t^{th} repeated trial model respectively. Let $f : w \to f(w)$ be the function that maps a weight w to its position in the ordered list. Using Equation 8, we found that, on average, a weight parameter moves by $4.62\% \pm 3.47\%$ from its original place in the ordered list with larger weights being more stationary compared to smaller weights.

$$\mathbb{E}_{w_i \in W^{ref}} \left[\mathbb{E}_{t \in \text{Trials}} \left[\frac{\left| f(w_i^{ref}) - f(w_i^t) \right|}{||W^{ref}||} \right] \right] \tag{8}$$

In order to improve the robustness of IMP towards the ordering of the training data and the influence of the stochastic gradient decent, we proposed and experimented with an augmented IMP algorithm. We augmented the iterative magnitude pruning algorithm with a scoring mechanism (IMPS) in order to centralize the weight magnitude distribution in the ordered list.

We trained K instances of the same model with different training data orderings which resulted in K different ordered parameter lists. For each model parameter w_i, its score was defined using Equation 9.

$$\text{Score}(w_i) = \frac{\sum_{k \in K} f\left(w_i^k\right)}{K} \tag{9}$$

In this definition, the score of each parameter is the observed expected value of its position over the K trained instances. The final ordered parameter list is obtained based on the score of each parameter. In Figure 4.2, an example of this process is shown for the weight magnitude distribution of three trained instances and the final ordered list that is obtained from them.

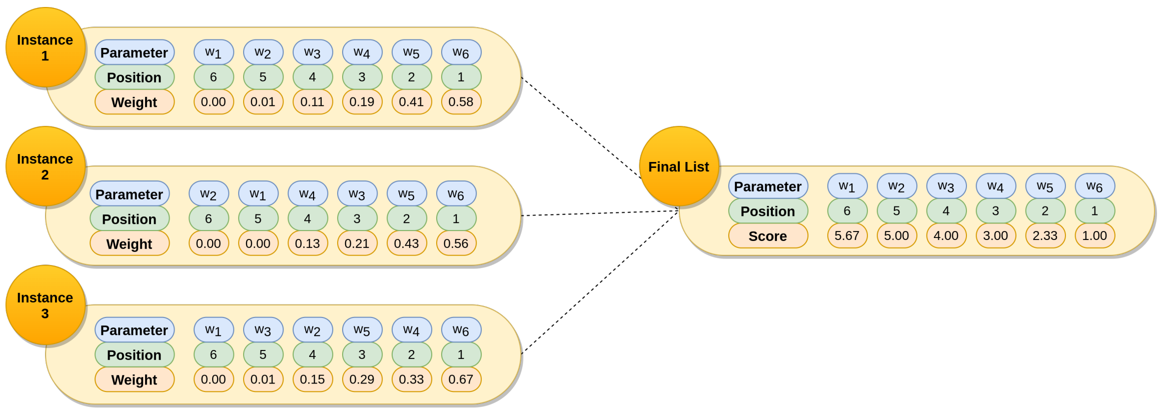


Figure 4.2: An example of parameter scoring in IMPS.

Although generally the law of large numbers entails a large value of K (repeated trials) in order to have convergence between the observed and the theoretical expected value of model parameters positions, due to limited computational powers, in all of our experiments with IMPS, we used five instances of the trained models to find the pruning parameter candidates. The details of these experiments are given in Sections 4.3.2.2 and 4.3.2.3.

4.3.2.2 TIMBERT With BERT Initialization

In this section we discuss the details of our experiments on TIMBER-Large-w/-Orthographic (see Table 4.1 row #4) in order to achieve a lighter model while preserving its performance. We experimented with parameter pruning using the IMP algorithm (Frankle and Carbin, May 2019), IMPS (see Section 4.3.2.1), and random pruning. In these experiments, the initialization and rewinding values of the backbone of the network are set to the values of the pretrained BERT-Large network. The initialization and rewinding values of the last layer of the network, the prediction layer, are set following the Glorot Initialization method (Glorot and Bengio, May 2010) and are kept the same for all pruning algorithms.

As shown in Figure 4.3.a, we can compress the model to extreme degrees with little loss in performance. Removing 60% of the model parameters results in only 0.8% loss of performance. Figure 4.3.b shows the performance gains of IMPS over the simple IMP. For most parts, these two approaches behave similarly and the issues discussed in Section 4.3.2.1 seem to have little to no effect on the performance of the pruned sub-networks. However, the impact of the observed

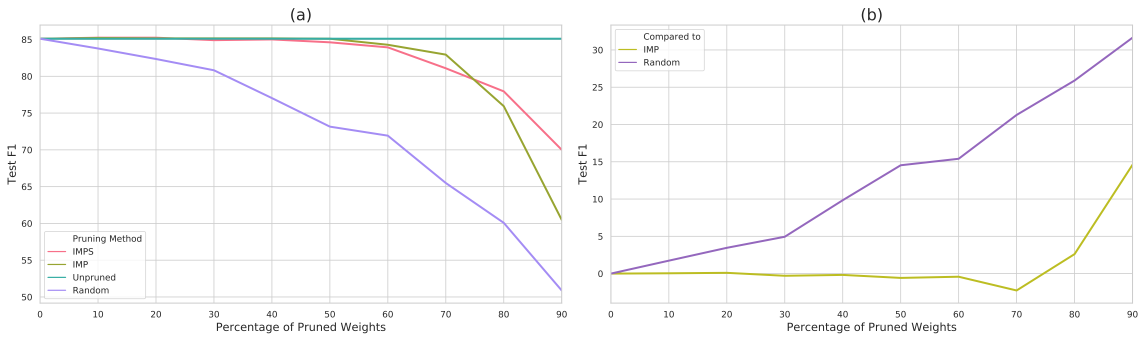


Figure 4.3: Parameter pruning on TIMBER-Large-w/-Orthographic with BERT-Large initialization values. (a) Shows the effect of different parameter pruning algorithms on performance. (b) Shows the relative performance gains of the IMPS algorithm compared to IMP and random pruning.

sensitives of IMP becomes visible in sever pruning regimes. At 80% and 90% pruning, IMPS outperforms IMP by 2.62% and 14.60% respectively, indicating the pruning stability of IMPS over IMP.

Our experiments with parameter pruning, could be seen as a permanent dropout which increases regularization and results in better generalization performance (Srivastava et al., June 2014). It can also be seen as a form of L_0 regularization since it encourages sparse model representations which has shown to improve generalization performance (Louizos et al., April 2018). Our experiments confirm these views; we found that the compressed TIMBER-Large-w/-Orthographic model via the IMP algorithm outperformed the uncompressed model at 10% pruning level with an F1 of 85.21%. The compressed model via IMPS algorithm outperformed the original model at pruning levels of 10% and 20% having F1 of 85.25%. Further details on the performance of the model obtained from the 20% pruning level is given in Table 4.1 (row #3).

4.3.2.3 TIMBERT With Toponym Identification Initialization

Previous work on the IMP algorithm (Frankle and Carbin, May 2019; Frankle et al., June 2019) has shown the importance of the network initialization and rewinding values in the stability and scalability of the Lottery Ticket Hypothesis. In the experiment of Section 4.3.2.2, the initialization and rewinding values came from a pre-trained language model designed to handle a variety of NLP tasks. In this section we investigate the effect of task specific initialization and rewinding values on the performance of the sub-networks derived from the IMP algorithm (Frankle and Carbin, May

2019), IMPS (see Section 4.3.2), and random pruning.

In order to fine-tune our model to a general domain toponym identifier, we used the CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, May 2003) which contains four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. We filtered the dataset to only include instances of location names in English and established a training set with 8.5k data points.

We fine-tuned our best performing model architecture i.e. TIMBER-Large-w/-Orthographic on this dataset and used early stopping to end the training process. The weights of this trained model constitute the initialization and rewinding values of our experiments in this section.

Using these initialization values, we further trained the network with the SemEval 2019 shared task 12 dataset (Weissenbacher et al., June 2019) and observed a significant improvement in the model performance, from 85.25% to 89.98% (see Table 4.1 row #3 and #2). We then repeated the same experiments described in Section 4.3.2.2 to observe the effect of task specific initialization on the performance of the pruned sub-models.

As shown in Figure 4.4.a, removing 60% of the network parameters results in negligible performance reduction (0.8% F1), which is consistent with our findings from Section 4.3.2.2. Figure 4.4.a also shows the robustness of IMPS and IMP in extreme levels of pruning. Figure 4.4.b, shows that the sub-networks derived from IMPS outperform those from IMP at all levels of pruning, although the performance gains are not significant at lower levels of pruning, we can clearly see the advantage of IMPS in extreme cases of pruning, outperforming IMP at 80% and 90% sparsity levels by 5.54% and 20.59% respectively. These findings, empirically confirm the stability and robustness of IMPS over the IMP algorithm.

We again observed that the compressed models outperformed the original uncompressed models. Compressing via IMP outperformed the original model at 10% and 20% sparsity levels with 90.18% and 90.08% F1 respectively. Compression via IMPS outperformed the original model at 10% and 20% pruning level with 90.85% and 90.37% F1 respectively. Further details on the best performing pruned model is given in Table 4.1 (row #1).

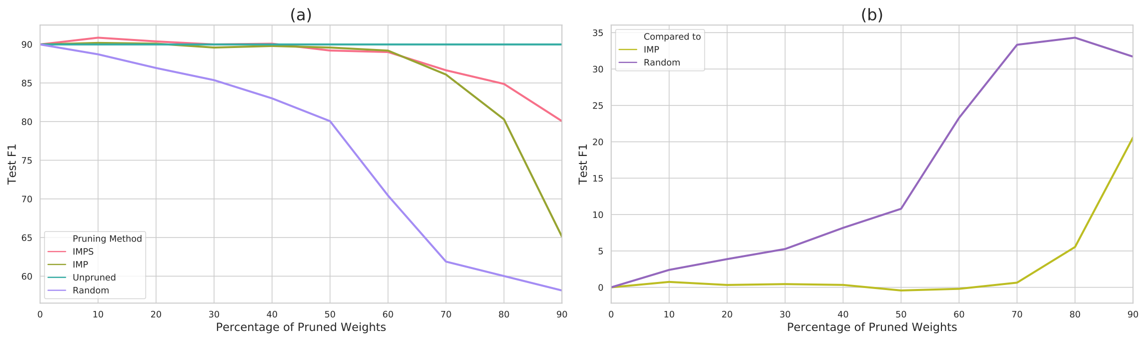


Figure 4.4: Parameter pruning on TIMBER-Large-w/-Orthographic with toponym identification initialization values. (a) Shows the effect of different parameter pruning algorithms on performance. (b) Shows the relative performance gains of the IMPS algorithm compared to IMP and random pruning.

4.4 Discussion

Our search for a toponym identifier for the medical domain with little to no task-specific design choices, led us to the development of the TIMBERT models. Our experiments with BERT as the backbone of our models detailed in Section 4.3.1 confirmed that certain linguistic insights such as POS tags are seamlessly transferred to downstream tasks while certain others such as Orthographic Features (see Section 4.3.1.1) need to be implemented as part of the model.

Our efforts to find a compact model while taking advantage of a memory and computationally intensive backbone model such as BERT led us to a recently proposed parameter pruning algorithm (Frankle and Carbin, May 2019), IMP, and we studied its application in a large scale NLP task. To the best of our knowledge, our study constitutes the first analysis of the impact of IMP in large scale NLP task. Our results of Section 4.3.2.2 showed that only a subset of the BERT model is needed to achieve competitive performance in a downstream task and the rest can be discarded with very little loss of performance.

Our experiments with IMP led to a novel algorithm, IMPS, that empirically is more stable and scalable in comparison to IMP. As shown in Section 4.3.2.2, sub-models produced from 90% pruning by IMP and IMPS lost 24.63% and 15.10% in performance respectively compared to the original unpruned models which confirms the superiority of IMPS. The same pattern emerged in our experiments in Section 4.3.2.3, when sub-models produced from 90% pruning by IMP and IMPS lost 24.87% and 9.92% respectively, indicating the robustness of IMPS.

Comparing our results from Sections 4.3.2.2 and 4.3.2.3 underlines the importance of network initialization in finding the winning tickets and suggests that task-specific fine-tuning could help to reduce the margin of performance loss. Furthermore, sub-models obtained from performing IMP and IMPS on a task specific backbone model achieved better performance in comparison to the ones gained from the general language model initialization.

4.5 Conclusion

In this chapter, we presented a compact and efficient model for toponym identification in the medical domain which significantly improves the state-of-the-art performance. Our approach successfully eliminated many design choices (e.g. network architecture and embedded linguistic features) and barriers (e.g. OOV and domain specific embeddings). Our search for the efficiency of the model led us to the development of a novel approach to parameter pruning and our results underlines the significance of our algorithm in severe pruning regimes.

Chapter 5

Conclusion and Future Work

5.1 Summary

This thesis explored the recognition of toponyms within the medical domain via neural networks. The goal of this thesis was to develop an automated mechanism for toponym detection within medical journals and reports. This task is beneficial to phylogeographers and epidemiologists who study and model the development and the global impact of the spread of viruses.

We trained and developed our models using the SemEval 2019 shared task 12 dataset (Weissenbacher et al., June 2019). The evaluation of the models were performed using the evaluation script provided by the SemEval 2019 shared task 12 organizers (Weissenbacher et al., June 2019) which reports the performance of the models on a predefined test set.

In Chapter 3, we presented and evaluated our first model: a deep feed-forward neural network (DFFNN) enriched with linguistic insights and domain specific embeddings. We chose the feed-forward architecture based on the hypothesis that localized contextual information is sufficient to identify the toponyms. When evaluated on the SemEval 2019 shared task 12 test set (Weissenbacher et al., June 2019), the DFFNN exceeded the baseline model's performance by 10.29% in F1 measure. These results showed the importance of domain specific word embeddings and certain linguistic features in identifying toponyms within the medical domain. Our results underlined the significance of knowledge transfer from domain specific word embeddings to downstream tasks within the medical domain.

In Chapter 4, our second attempt at toponym detection, we were motivated to experiment with neural architectures that considered all available contextual information (Vaswani et al., January 2017; Lan et al., September 2019) within a meaningful unit of language. We used the BERT model (Devlin et al., June 2019) as the backbone of our toponym identifier system. We created a family of Toponym Identification Models based on BERT (which we call TIMBERT) as the core of the network, in order to learn directly in an end-to-end fashion the mapping from the input sentence to the associated output labels. When evaluated with the SemEval 2019 shared task 12 test set (Weissenbacher et al., June 2019), our best model achieved an F1 score of 90.85%, a significant improvement compared to our DFNN model's performance of 80.13% and the state-of-the-art of 89.10% (Wang et al., June 2019; Magnusson and Dietz, June 2019).

The significant performance enhancement of our BERT based model came at a cost of memory and computation complexity. In order to reduce the computation costs of TIMBERT we experimented with parameter pruning and proposed a novel algorithm, IMPS, for large scale parameter pruning of neural networks, then investigated its application in the toponym recognition task.

Our experiments with parameter pruning resulted in compressed TIMBERT models that consistently outperformed the uncompressed models at 10% pruning level. Extreme regimes of pruning, such as 90% parameter reduction, via our proposed IMPS algorithm led to minimal loss of performance which underlines the significant ability of our pruning algorithm in finding the less essential structures within a neural network and removing them.

In Chapter 4, we also examined the effect of parameter pruning in the transfer learning domain. Our results showed the importance of network initialization values in finding the optimal compressed neural network from an uncompressed base model and suggested that task-specific fine-tuning of the base model prior to its knowledge transfer to a downstream task could help to reduce the margin of performance loss.

5.2 Contributions

This thesis presented a number of theoretical and practical contributions, including:

- the implementation of different feed-forward neural networks enriched with linguistic insights for toponym identification within the medical domain for the SemEval 2019 shared task 12 (Weissenbacher et al., June 2019). This led to two publications: Davari et al. (April 2019) and Davari et al. (to appear 2020). (see Chapter 3)
- a novel approach to toponym detection in the medical domain, based on knowledge transfer from language models which achieves the state-of-the-art performance on the SemEval 2019 shared task 12 test set (Weissenbacher et al., June 2019). (see Chapter 4)
- a novel parameter pruning algorithm, based on the Lottery Ticket Hypothesis (Frankle and Carbin, May 2019) algorithm, for large scale neural networks which leads to minimal loss of performance even in severe pruning regimes. (see Chapter 4)
- insights into the benefits of tasks-specific fine-tuning of large scale language models prior to their knowledge transfer to downstream tasks. (see Chapter 4)

5.3 Future Work

In Chapter 3, we developed a toponym detection model and showed that the presence of certain linguistic features improve the model performance. More specifically, we investigated the effect of: removal of punctuation marks, removal of stop words, orthographic features, part of speech tags, and lemmatization. An in-depth investigation of the influence of other linguistic features that we did not examine, would reveal further insights on the task of toponym detection. As a result, future neural models could leverage these insights to enhance their performance.

In Chapter 3, our experiments with deeper neural models were fruitless due to the small size of the dataset. The models could be extended for better performance provided more human annotated data. However, since human annotated data is expensive to produce, we suggest distant supervision (Krause et al., November 2012) to be explored to further improve results in the task. Furthermore, in Section 4.3.2.3, we saw the benefits of task-specific transfer learning. It would be interesting to see whether task-specific transfer learning could further improve the performance of the model we developed in Chapter 3.

In Chapter 4, we developed a family of Toponym Identification Models based on BERT (TIMBERT) as the backbone of the network. In future studies, we would like to investigate the effects of other language models, such as XLNET (Yang et al., December 2019), RoBERTa (Liu et al., July 2019), and ALBERT (Lan et al., September 2019), for the backbone of TIMBERT and study the behaviour of parameter pruning algorithms on these models.

In Chapter 4, we proposed a novel iterative parameter pruning algorithm (IMPS), based on the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin, May 2019) algorithm, augmented with a scoring mechanism to centralize parameter ranking. We applied our algorithm in the domain of transfer learning to a large scale natural language processing (NLP) task and found empirical evidence of stability and robustness of IMPS over the LTH algorithm.

Future studies on other NLP tasks is needed in order to confirm the performance gains of IMPS over LTH that we observed for the toponym identification task. Moreover, we would also like to see whether this pattern continues in other domains that leverage transfer learning such as computer vision.

An additional research direction is the performance and stability analysis of IMPS compared to LTH when applied to neural networks for which the learning process does not involve transfer learning. For this specific setting, Frankle et al. (June 2019) (see Section 2.11) have introduced a more stable and scalable variation of LTH. The main idea of our algorithm could easily be integrated with this work and it would be interesting to see whether any performance improvement can be achieved.

Overall, the main objective of this thesis was to make use of the current developments in the field of deep learning and natural language processing (NLP) to further facilitate the automation of text based processes in the biomedical domain. NLP in the biomedical domain faces many challenges such as: the small number of publicly available datasets, the relatively small training samples per dataset, the domain specific vocabulary, etc. Considering the problems surrounding the field of NLP in the biomedical domain and the relatively recent interest in applying neural network techniques to bridge the gap between model performance in the general English domain and biomedical domain, much more research still remains.

Bibliography

Einat Amitay, Nadav Har'El, Ron Sivan, and Aya Soffer. Web-a-where: geotagging web content. In *Proceedings of the 27th International ACM Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 273–280, Sheffield, UK, July 2004. ACM.

Mariona Coll Ardanuy and Caroline Sporleder. Toponym disambiguation in historical documents using semantic and geographic features. In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, pages 175–180. ACM, June 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, July 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, USA, May 2015.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155, February 2003.

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, March 1994.

Dennis A Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic Acids Research*, 41(D1):D36–D42, November 2012.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., June 2009.

Phillipe Boeuf, Heidi E Drummer, Jack S Richards, Michelle JL Scoullar, and James G Beeson. The global threat of zika virus to pregnancy: epidemiology, clinical perspectives, mechanisms, and impact. *BMC medicine*, 14(1):1–9, August 2016.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 1442–1451, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

John A Bullinaria and Joseph P Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526, August 2007.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 2358–2367, Berlin, Germany, August 2016. Association for Computational Linguistics.

Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370, July 2016.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734, Doha, Qatar, October 2014.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (ACL 2019)*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 160–167, Helsinki, Finland, July 2008. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, August 2011.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of The 24th International Conference on Machine Learning (ICML 2007)*, pages 193–200, Corvallis, USA, June 2007. ACM.

MohammadReza Davari, Leila Kosseim, and Tien D Bui. Toponym identification in epidemiology articles–a deep learning approach. In *Proceedings of The 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2019)*, La Rochelle, France, April 2019.

MohammadReza Davari, Leila Kosseim, and Tien D Bui. Toponym mention identification in biomedical texts. In *Biomedical Literature Mining: Methods and Protocols*. Springer Nature, to appear 2020.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, March 1990.

Grant DeLozier, Jason Baldridge, and Loretta London. Gazetteer-independent toponym resolution using geographic word profiles. In *Proceedings of 29th Annual Conference of Association for the Advancement of Artificial Intelligence (AAAI 2015)*, pages 2382–2388, Austin, USA, February 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2019)*, Minneapolis, USA, June 2019.

Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, October 2019.

W Nelson Francis and Henry Kucera. Brown corpus. *Department of Linguistics, Brown University, Providence, Rhode Island*, 1, 1964.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *The 7th International Conference on Learning Representations (ICLR 2019)*, New Orleans, USA, May 2019.

Jonathan Frankle, G Karolina Dziugaite, DM Roy, and M Carbin. Stabilizing the lottery ticket hypothesis. *arXiv, page*, June 2019.

Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, February 1994.

Eric Garbin and Inderjeet Mani. Disambiguating toponyms in news. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, pages 363–370, Vancouver, Canada, October 2005. Association for Computational Linguistics.

P. Gautret, E. Botelho-Nevers, P. Brouqui, and P. Parola. The spread of vaccine-preventable diseases by international travellers: A public-health concern. *Clinical Microbiology and Infection*, 18:77 – 84, October 2012. ISSN 1198-743X.

Judith Gelernter and Shilpa Balaji. An algorithm for local geoparsing of microtext. *GeoInformatica*, 17(4):635–667, January 2013.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pages 249–256, Sardinia, Italy, May 2010.

Google. Pretrained word and phrase vectors. `https://code.google.com/archive/p/word2vec/`, 2019. Accessed: 2019-01-10.

Juliana Grant, Aaron M Wendelboe, Arthur Wendel, Barbara Jepson, Paul Torres, Chad Smelser, and Robert T Rolfs. Spinach-associated escherichia coli o157: H7 outbreak, utah and new mexico, 2006. *Emerging infectious diseases*, 14(10):1633, October 2008.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, December 2014.

A. D. Green and K. I. Roberts. Recent trends in infectious diseases for travellers. *Occupational Medicine*, 50(8):560–565, November 2000. doi: 10.1093/occmed/50.8.560. URL `http://dx.doi.org/10.1093/occmed/50.8.560`.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS 2015)*, pages 1135–1143, Montreal, Canada, December 2015.

Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems (NIPS 1993)*, pages 164–171, Denver, USA, December 1993.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 770–778, Las Vegas, USA, June 2016.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, July 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, November 1997.

James E Hoffman and Baskaran Subramaniam. The role of visual attention in saccadic eye movements. *Perception & psychophysics*, 57(6):787–795, January 1995.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

W. Kienreich, M. Granitzer, and M. Lux. Geospatial anchoring of encyclopedia articles. In *Proceedings of the 10th International Conference on Information Visualisation (IV 2006)*, pages 211–215, London, UK, July 2006.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the 13th Conference of Association for the Advancement of Artificial Intelligence (AAAI 2016)*, AAAI'16, pages 2741–2749, Phoenix, USA, February 2016. AAAI Press.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems (NIPS 2015)*, pages 3294–3302. Curran Associates, Inc., Montreal, Canada, December 2015.

Sebastian Krause, Hong Li, Hans Uszkoreit, and Feiyu Xu. Large-scale learning of relation-extraction rules with distant supervision from the web. In *Proceedings of The 11th International Semantic Web Conference (ISWC 2012)*, pages 263–278, Boston, USA, November 2012. Springer.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*, pages 1097–1105, Lake Tahoe, USA, December 2012.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of The 15th Annual Conference of The North American Chapter of The Association for Computational Linguistics (HLT-NAACL 2016)*, San Diego, USA, June 2016.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, September 2019.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of The 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, Paris, France, May 2010. IEEE.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NIPS 1990)*, pages 598–605, Denver, USA, November 1990.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, October 1999.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *The 5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France, April 2017.

Lishuang Li, Liuke Jin, Zhenchao Jiang, Dingxin Song, and Degen Huang. Biomedical named entity recognition based on extended recurrent neural networks. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2015)*, pages 649–652, Washington, USA, November 2015.

Michael D Lieberman and Hanan Samet. Multifaceted toponym recognition for streaming news. In *Proceedings of the 34th International ACM Conference on Research and Development in Information Retrieval (SIGIR 2011)*, San Francisco, USA, December 2011. ACM.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, July 2019.

Marc Moreno Lopez and Jugal Kalita. Deep learning applied to NLP. *arXiv preprint arXiv:1703.03091*, March 2017.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada, April 2018.

Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208, June 1996.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1412–1421, Lisbon, Portugal, September 2015.

Arjun Magge, Davy Weissenbacher, Abeed Sarker, Matthew Scotch, and Graciela Gonzalez-Hernandez. Deep neural networks and distant supervision for geographic location mention extraction. *Bioinformatics*, 34(13):i565–i573, July 2018.

Matthew Magnusson and Laura Dietz. UNH at SemEval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 1308–1312, Minneapolis, USA, June 2019. Association for Computational Linguistics.

R Mandavia, K Kapoor, J Ouyang, and H Osmani. Evaluating ear cartilage piercing practices in London, UK. *The Journal of Laryngology & Otology*, 128(6):508–511, June 2014.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June 1993.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations (ICLR 2013)*, Scottsdale, USA, May 2013.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, September 2013.

SPFGH Moen and Tapio Salakoski2 Sophia Ananiadou. Distributional semantics resources for biomedical text processing. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine (LBM 2013)*, pages 39–43, Tokyo, Japan, December 2013.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019)*, pages 11264–11272, Long Beach, USA, June 2019.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 479–489, Austin, Texas, November 2016. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, January 2007.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)*, pages 1717–1724, Columbus, USA, June 2014.

Sankar K Pal and Sushmita Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on neural networks*, 3(5):683–697, September 1992.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML 2013)*, pages 1310–1318, Atlanta, USA, June 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/, 2014. Accessed: 2019-01-10.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, October 2014.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (NAACL 2018)*, pages 2227–2237, New Orleans, USA, June 2018. Association for Computational Linguistics.

Ross S Purves, Paul Clough, Christopher B Jones, Avi Arampatzis, Benedicte Bucher, David Finch, Gaihua Fu, Hideo Joho, Awase Khirni Syed, Subodh Vaid, et al. The design and implementation of spirit: a spatially aware search engine for information retrieval on the internet. *International journal of geographical information science*, 21(7):717–745, June 2007.

Teng Qin, Rong Xiao, Lei Fang, Xing Xie, and Lei Zhang. An efficient location extraction algorithm by leveraging web contextual information. In *Proceedings of the 18th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2010)*, pages 53–60, San Jose, USA, November 2010. ACM.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, June 2018.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, May 2017.

João Santos, Ivo Anastácio, and Bruno Martins. Using machine learning methods for disambiguating place references in textual documents. *GeoJournal*, 80(3):375–392, June 2015.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, September 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, June 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS 2014)*, pages 3104–3112, Montreal, Canada, December 2014.

Javier Tamames and Victor de Lorenzo. Envmine: A text-mining system for the automatic extraction of contextual information. *BMC bioinformatics*, 11(1):294, June 2010.

Mike Taylor. *Reduced Geographic Scope as a Strategy for Toponym Resolution*. PhD thesis, Northern Arizona University, December 2017.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning at (HLT-NAACL 2003)*, pages 142–147, Edmonton, Canada, May 2003.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS 2017)*, pages 5998–6008, Long Beach, USA, January 2017.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (AMTA 2018)*, pages 193–199, Boston, MA, March 2018. Association for Machine Translation in the Americas.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems (NIPS 2015)*, pages 2773–2781, Montreal, Canada, December 2015.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Faqiang Wang, Wangmeng Zuo, Liang Lin, David Zhang, and Lei Zhang. Joint learning of single-image and cross-image representations for person re-identification. In *Proceedings of The 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 1288–1296, Las Vegas, USA, June 2016.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*, November 2015.

Xiaobin Wang, Chunping Ma, Huafei Zheng, Chu Liu, Pengjun Xie, Linlin Li, and Luo Si. DM_NLP at SemEval-2018 task 12: A pipeline system for toponym resolution. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 917–923, Minneapolis, USA, June 2019. Association for Computational Linguistics.

Davy Weissenbacher, Tasnia Tahsin, Rachel Beard, Mari Figaro, Robert Rivera, Matthew Scotch, and Graciela Gonzalez. Knowledge-driven geospatial location resolution for phylogeographic models of virus migration. *Bioinformatics*, 31(12):i348–i356, June 2015.

Davy Weissenbacher, Arjun Magge, Karen O'Connor, Matthew Scotch, and Graciela Gonzalez-Hernandez. SemEval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, Minneapolis, USA, June 2019. Association for Computational Linguistics.

Davy Weissenbacher, Abeed Sarker, Tasnia Tahsin, Matthew Scotch, and Graciela Gonzalez. Extracting geographic locations from the literature for virus phylogeography using supervised and distant supervision methods. *Proceedings of the AMIA Summits on Translational Science*, 2017: 114, November 2017.

World Health Organization. Factors that contributed to undetected spread of the ebola virus and impeded rapid containment, January 2015. URL https://www.who.int/csr/disease/ebola/one-year-report/factors/en/.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, September 2016.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, volume 37, pages 2048–2057, Lille, France, July 2015.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, volume 1, pages 1237–1247, Vancouver, Canada, July 2017.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *The 33rd annual conference in Advances in Neural Information Processing Systems (NIPS 2019)*, pages 5753–5763. Curran Associates, Inc., Vancouver, Canada, December 2019.