THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

PSHAT - Part of Speech Handling for Aramaic in the Talmud

by

Noah Santacruz

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Engineering

April 20, 2017

Advisor

Professor Carl Sable

THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

This thesis was prepared under the direction of the Candidate's Thesis Advisor and has received approval. It was submitted to the Dean of the School of Engineering and the full Faculty, and was approved as partial fulfillment of the requirements for the degree of Master of Engineering.

Professor Richard Stock
Acting Dean, School of Engineering

Professor Carl Sable
Candidate's Thesis Advisor

Abstract

This thesis deals with POS tagging in the Talmud. We have created the PSHAT system, which handles the lack of punctuation and a language mixture of Hebrew and Aramaic in the Talmud. PSHAT employs two bidirectional long short-term memory networks, one to tag each word by language and another to tag Aramaic words with their part-of-speech. We have validated that our system is robust for varying mixtures of Hebrew and Aramaic and that it improves performance on unknown words compared to a baseline. PSHAT has been implemented such that it could be easily applied to other mixtures of languages as well. The code and data used for this project can be found at `https://github.com/nsantacruz/PSHAT`.

Acknowledgments

I would like to thank those whose help made this thesis possible. First and foremost, I would like to thank Professor Carl Sable, my advisor. He went above and beyond what was expected of him, by advising me throughout the process while I was abroad. His advice was invaluable for the completion of this thesis.

I also want to thank Avi Shmidman and Moshe Koppel who are part of the Dicta research group in Israel. They shared their expertise in Hebrew NLP which formed the basis for this project.

Steve Kaufman, from the Comprehensive Aramaic Lexicon, graciously provided the Aramaic POS dataset used. In addition, he took the time to explain nuances in the data. I thank him for his help. In addition, I want to thank Sefaria, for creating an open-source library which made it very easy to get the Talmud text used in this thesis. Also, I want to thank them for generally being amazing.

Finally, I would like to thank G-d, without whom the many pieces need to complete such a project would not have fallen into place.

Contents

List of Figures

Chapter 1

Introduction

Syntactical analysis of text has applications in many areas of natural language processing (NLP). These include machine translation [31], sentiment analysis [2] and named-entity recognition [37]. A specific type of syntactic analysis often used is part-of-speech (POS) tagging, which associates a syntactic category with each word in a phrase or sentence. This is often non-trivial because of the ambiguity of many words when seen out of context.

POS tagging has been heavily studied for modern texts, such as microblogs [13], news articles [5] and reviews [24]. In the past, hidden Markov models (HMMs) have proven effective for POS tagging. Recently, neural networks have beaten HMMs at POS tagging. These include recurrent neural networks (RNNs) [43] and long short-term memory (LSTM) networks [52]. The last method has proven especially powerful at learning patterns over long sequences, something that HMMs cannot do.

However, there is still significant work in order to achieve results for ancient texts that are on par with results for modern documents. These texts pose unique challenges not present in modern contexts. This thesis focuses on POS tagging methods for the

Talmud. There are two features of the Talmud which make it particularly difficult for current POS tagging algorithms; the text contains no punctuation to seperate ideas or sentences and the text contains a mixture of multiple distinct languages.

POS taggers, whether based on HMMs or on neural networks, operate on sentences or short phrases. They also tend to operate on texts with only one language, although there are exceptions. We adapt current state-of-the-art POS tagging methods to work when these two conditions don't hold. Our approach involves a two-stage classifier. The first stage segments the Talmud into its two constituent languages, Hebrew and Aramaic. The second uses an LSTM paired with a multi-layer perceptron (MLP) network as the POS tagger. We have found that LSTMs are successful at learning sentence structure without explicit segmentation of the text. Additionally, the LSTM is robust even when dealing with sentences made up of Hebrew and Aramaic. This paper specifically focuses on tagging the Aramaic in the Talmud, although it sets up a framework which simplifies tagging the Hebrew as well. The dataset used was graciously provided by the Comprehensive Aramaic Lexicon (CAL) [27].

Chapter 2

Part-of-Speech Tagging

2.1 Overview

Part-of-speech tagging is the process of marking the part-of-speech (POS) for words in a document. A POS is a description of the syntactic usage of a word in a phrase or sentence. Examples of POS which exist in most languages are nouns, adjectives and verbs. Humans who are familiar with the grammar rules of a certain language can achieve high accuracy when tagging a text manually. Based on this, researchers believe that computers can achieve comparable results by mimicking and improving on the process used by humans.

There are two major challenges in POS tagging. The first is ambiguous words and the second is ambiguous sentences. Ambiguous words are words that can have multiple POS tags, depending on context. A simple example is:

```
I book the ticket.
```

The word 'book' can be either a verb or a noun, but in this sentence it must be a

verb acting on the noun 'ticket'. Ambiguous sentences are sentences that have more than one possible set of POS tags. For example:

```
Time flies like an arrow.
```

Two possible interpretations of this sentence are:

1. A command. You should time flies in the way you would time an arrow.

2. A statement. Time moves fast just like arrows move fast.

From the perspective of a POS tagger, the difference is whether 'time' is a verb and 'flies' is a noun (option 1) or 'time' is a noun and 'flies' is a verb (option 2). To correctly parse this sentence, a POS tagger must understand that option 2 is much more likely to be written than option 1.

English has 9 basic parts-of-speech [33]; namely, noun, pronoun, verb, adverb, adjective, conjuction, preposition, article and interjection. Every word in English can be categorized using one of these POS. However, much more specific POS can be defined just as easily. For example nouns can be broken into singular nouns, plural nouns, proper singular nouns and proper plural nouns. The level of detail used for POS depends on the intended use.

A paricular set of POS is known as a tag set. Researchers in NLP tend to use large tag sets. However, there has not been consensus on exactly which POS to include and how specific the definitions should be. For example for English, the Penn Treebank defines 36 POS tags [42] while the Brown Corpus defines 226 POS tags [51]. As [3] discusses, a common reason why tag sets can differ so much is that researchers creating these tag sets don't know in advance the applications they'll be used for.

Therefore, they tend to use their own judgement as to how specific the POS should be.

Tag sets have been used to manually tag corpora of text. These copora, referred to as treebanks, are generally used for studying the syntactic structure of a given set of text. Some copora contain very diverse texts; for example the Brown Corpus contains 15 different genres of text [25]. Others contain more focused text; for example the Penn Treebank is a collection of stories from the Wall Street Journal. These treebanks need to be tagged by multiple experts in syntax because there are often cases where the exact tag for a word is difficult to define.

Treebanks are necessarily language-specific since they are tagged based on the specific syntax of a language. Many treebanks have been created for English. However, treebanks for other languages vary depending on the popularity of the language. Unsurprisingly, there are very few treebanks available for Talmudic Aramaic. The only one we are aware of is the Comprehensive Aramaic Lexicon [27], which we have used for this project.

2.2 Rule-based POS Taggers

Because of these two challenges, automatic POS taggers must incorporate context into their algorithms. Early solutions used a ruled-based systems [30]. This involved creating a dictionary mapping every word to its possible POS. When tagging, if more than one possible POS exists for a word, basic syntactic rules are applied which take into account the POS of the previous and next words. This approach however requires expert knowledge of the language to write all of the grammar rules. Additionally, the system would need to be periodically updated to account for neologisms and proper

nouns.

2.3 Hidden Markov Model POS Taggers

Up until recently, the state-of-the-art approaches for POS tagging have used hidden Markov models [43]. The motivation for applying HMMs to POS tagging is that often times it is possible to disambiguate a word's POS just by looking at the previous 1 or 2 words in the sentence. k^{th}-order HMMs are based off of the assumption that the probability of a given state is dependent only on the previous k states and the current observed token. Concretely, this probability is:

$$\underset{s}{\operatorname{argmax}}\, y_i = P(x_i|s)P(s|y_{i-1}, y_{i-2}, ..., y_{i-k}) \qquad s \in S \tag{2.1}$$

where $x_1, ..., x_i$ are the observed tokens, and $y_1, ..., y_i$ are the corresponding hidden states. S is the set of possible hidden states. In order to implement this approach, an HMM requires a probability table containing the probability of any state given the previous k states. Additionally, it needs a table for the probability for every observed token given a hidden state. This amounts to approximately $N^{k+1} + N \times M$ parameters where N and M are the number of hidden and observed states respectively and k is the order of the HMM. In the case of POS tagging, the observed tokens are the words in a sentence and the hidden states are the parts of speech. Assuming a vocabulary of 10,000 words with 40 POS tags and a 2nd order HMM, this leads to almost half a million parameters.

A very large dataset is needed to properly estimate these parameters. Because many of the POS combinations in this table never occur, approximation algorithms

have been used to trim the parameter space [12]. For Modern English, HMM-based taggers tend to achieve about 96% accuracy [5].

2.4 Neural Networks

Recently, neural network approaches have surpassed HMMs for POS tagging. The most successful neural network used for POS tagging has been the bidirectional long short-term memory network (BiLSTM) [53]. This network can learn patterns in an arbitrarily long sequence. BiLSTMs have two major advantages over HMMs. The first is that they can use context both before and afterwards when deciding the POS for a word. The second is that the size of the context is not fixed; it depends on the current word and the context around that word. This allows BiLSTMs to learn more flexible models when modeling the syntax of a language. An overview of neural networks will be given in the next chapter.

Chapter 3

Neural Network

3.1 Feedforward Networks

Neural networks were initially inspired by the interconnectedness of neurons in the brain. Feedforward neural networks are the most basic type of neural network. An example feedforward network is shown in Figure 3.1.

This network contains 4 layers: an input layer, two hidden layers and an output layer. $x_0 ... x_3$ represent the 4 inputs to the network, which make up the input layer. The input to each of the nodes in the hidden and output layers is a weighted sum of all the nodes in the previous layer plus a bias input, shown below the network. The weights for each edge in the network are defined by a weight matrix. A sample weight is shown for the edge between the first node in the input layer and the first node in hidden layer 1. Then, a nonlinear function is applied to the sum. This process continues until the output layer. For classification tasks with mutually exclusive categories, such as POS tagging, the number of output nodes corresponds to the number of classes. The value of the nodes in the output layer is then viewed as an unnormalized probability distribution and the one with the maximum value is chosen

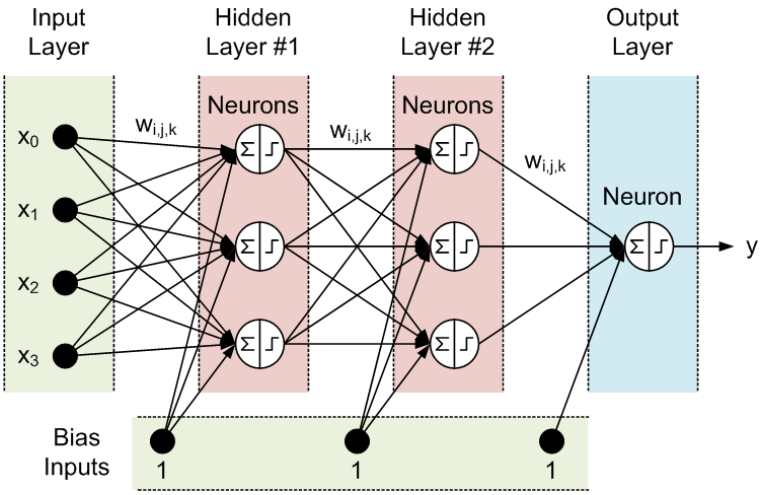

Figure 3.1 A sample feedforward network with 4 layers [38]

as the predicted class.

Mathematically, the layers of a feedforward network are vectors and the weights of the edges between two layers are matrices. The value of the j^{th} layer is then

$$x_j = f(W_{(j-1,j)}x_{j-1} + b) \tag{3.1}$$

where x_j is of size N and x_{j-1} is of size M, $W_{(j-1,j)}$ is the weight matrix of size $N \times M$ and b is the bias vector of size N. f is a nonlinear function.

Training of the network involves passing training vectors into the input layer. These are forward propagated using the above equation until they reach the output layer. The output vector is compared to the known class corresponding to the training vector, giving the loss. The gradient of the loss is used to update the weights of the network using an algorithm known as backpropagation. One iteration over the training data is referred to as an epoch. In order to achieve a global minimum of the loss over all training examples, the network is trained over many epochs until some stopping criterion is met [14].

There are two issues with using a standard feedforward network for NLP tasks. The first is that neural networks require numerical input. The second is that a single word passed into the network contains no context.

To solve the first issue, researchers have used embedding tables. These are lookup tables, where every token in the input vocabulary is matched to a corresponding vector. Initially, these vectors are chosen randomly. They are then updated as if they were the first layer in the network, using the backpropagation algorithm. Embedding vectors are especially useful in NLP tasks because they allow the model to generalize to words that are not well represented in the training set. For example, a POS tagger may come across the word 'undulating' which appears rarely in the training set. However, it has seen many examples of verbs ending in '-ing'. The embedding vector for 'undulating' might therefore be close to other '-ing' verbs in which case it will be more likely to know that it is a verb [15].

3.2 Recurrent Neural Networks

The issue of context is solved by a recurrent neural network (RNN) [16]. RNNs can be viewed as a function of the form:

$$Y_n = RNN(X_1 : n) \tag{3.2}$$

where $X_1 : n$ represents n d-dimensional vectors and Y_n is the output vector. Since n can vary, the RNN function can be viewed as a lossy compression function.

Figure 3.2 represents the dependence on previous inputs of an RNN over several time steps. Every column represents an architecture similar to a feedforward network.

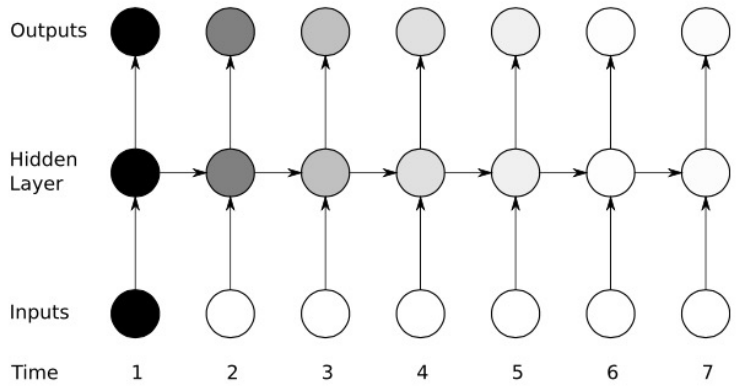

Figure 3.2 An RNN showing the dependence on previous inputs [55]

The main difference is that adjacent columns are connected. The darkness of a column represents the effect of the input at time 1. In each consecutive time step, this effect diminishes. Effectively this means the output at time t is dependent on all previous inputs.

A simple implementation of an RNN (S-RNN) is where the input to a certain hidden layer is the concatenation of the input layer and the previous hidden layer in time. However, it can be unpredictable to train an S-RNN, because the gradient tends to vanish or explode.

The long short-term memory (LSTM) is intended to overcome this problem [17]. LSTM networks have a state s. This state is stored in the hidden layer and has 2 components. The first is the memory of the network c, and the second is the hidden state, h, which is the equivalent of the hidden layer in a feedforward network. The network has three types of gates to control access to this state: forget gates, input gates and output gates. Forget gates control how the memory is updated each time step. Input gates control how much input is passed into the network. Output gates control how the hidden state is updated. Each one of these gates has weights which

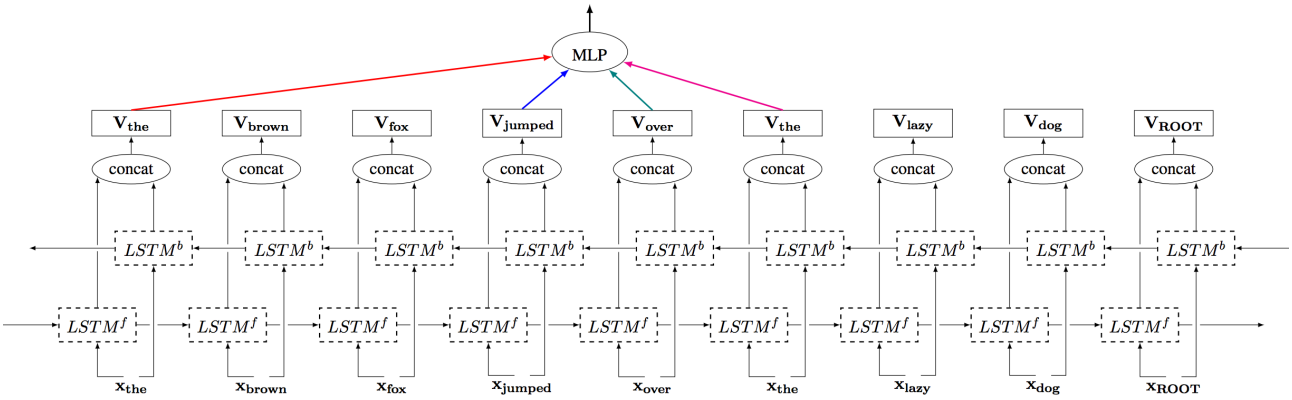

Figure 3.3 An example of the way a BiLSTM handles input [29]

are learned using an algorithm similar to backpropagation. It has been found that LSTMs can learn very complex sequences without the issue vanishing / exploding gradients.

As opposed to LSTMs which only save past context, bidirectional LSTMs (BiLSTMs) can account for the full context around a given input vector. This is useful in cases where the whole input is known from the beginning, such as POS tagging and many other NLP tasks. BiLSTMs are simply two LSTMs with their outputs connected.

Figure 3.3 above shows the architecture of BiLSTM. At the bottom, the input is being fed into both LSTMs. The lower LSTM is reading the sentence forwards while the upper one is reading the sentence backwards, as indicated by the horizontal arrows of each. In practice, the input sequence is passed into the forward LSTM and the output for each input vector is saved in a list. The sequence is then passed backwards into the backwards LSTM. The final vector corresponding to each input word is the concatenation of the output from the forward LSTM and the backward LSTM. Concatenation here means appending the elements of one vector to the other. This effectively gives each input both the forward and backward contexts. These

vectors are shown on top of the 'concat' operations. Each vector is then indivually passed into an multi-layer perceptron network (MLP a.k.a. feedforward network) which classifies the words [18].

3.3 Beam Search

While not strictly a neural network algorithm, beam search has proven very useful for training neural networks on sequence learning tasks such as POS tagging. Beam search is an approximation of breadth-first search, useful when a tree is too large to search completely [11]. Instead of expanding all nodes at a given level in the tree, beam search only expands the N best nodes. Therefore, the search space at any level is always constant. A heuristic is necessary in order to rate the nodes.

[8] shows how beam search is useful for training neural networks for sequence learning tasks. Taking POS tagging as an example, the first word of a sentence is read and the probability distribution for the potential POS tags is generated. This can be taken from the softmax output of a feedforward network. The next word is read and the POS probability distribution is generated. Multiplying the probability for each of the potential POS by the probability of the N POS tags currently in the beam, the combined probability is calculated. The N sequences with the highest probability are chosen to remain in the beam. This continues until the end of the sentence. The final sequence will be the one with highest combined probability.

[8] discusses the idea of early update when applying beam search to neural networks. In case the correct POS for a certain word no longer exists in the beam, the network stops parsing the sentence and goes straight to calculating the loss. Beyond saving time, this prevents the network from overfitting to examples in the training set.

Chapter 4

Talmud

4.1 Overview

The Talmud is a work composed over hundreds of years (70 CE - 5th century) which contains a comprehensive discussion of Jewish law. It is composed of two distinct parts, the Mishnah and the Gemara. The Mishnah was the first complete work of Jewish law composed. It is made up of 63 books, referred to as tractates, which each cover a specific area of law. Before it, law was passed orally from rabbi to student. However, after the destruction of the second temple at the beginning of the Common Era, this model of transmission of Jewish law became untenable, spurring the writing of the Mishnah [20]. Due to the long span of time when it remained oral, the Mishnah does not always present a final ruling; it often cites disagreements among Rabbis as to what the law should be in a particular circumstance. The Gemara is generally a later discussion of topics covered in the Mishnah. Many times, the Gemara tries to reconcile disagreements and contradictions found in the Mishnah. However it also discusses tangential topics in philosophy, mythology, ethics, history and mysticism. There are two versions of Talmud, the Palestinian Talmud and the Babylonian Talmud. The

Figure 4.1 The layout of the Mishnah and the Gemara in the Vilna edition of the Talmud

Babylonian Talmud is studied far more and has much more significance to modern Jewish law [50]; it will be referred to throughout this paper simply as the Talmud.

While there exist many textual variants of the Talmud, the Vilna edition, published in 1880, is by far the most popular. It follows the pagination of the Bomburg edition, which was the first complete printing of the Talmud. Using this pagination, there are 5,894 pages. Nowadays, all references to Talmud use this numbering system [48].

4.2 Language

The textual relationship between the Gemara and Mishnah is partially structured and partially unstructured. Figure 4.1 shows an example of how a page of the Talmud is laid out. The Talmud is generally structured by quoting a paragraph of Mishnah, shown in blue, followed by the Gemara commentary on that paragraph of the Mishnah, shown in red. However, topics discussed in a paragraph of the Mishnah are often related to other Mishnayot (plural for paragraphs of the Mishnah in Hebrew). The Gemara will mention these related Mishnayot without any citation and will usu-

ally only quote a small portion from the text. An example of this is from Tractate Berakhot 2a:

> **ותו מאי שנא דתני בערבית ברישא לתני דשחרית ברישא**
>
> And furthermore, why does it [the Mishnah] teach 'in the evening' first? Let it teach 'in the morning' first. [46]

The Hebrew words in this passage have been underlined. The passage is discussing the exact language of the Mishnah. Because it is quoting and ammending the Mishnah, it uses the Hebrew words **ערבית** and **שחרית**.

It is also common for the Gemara to quote from other works extant around the time of the Mishnah. These include the Baraita, the Tosefta and the Midrash. In general, these works are referred to as the Tannaitic, referring to the title given to rabbis living around the time of the Mishnah. An example is of this from Tractate Rosh Hashanah 13b is:

> **מי לא תניא רבי יוסי בן כיפר אמר משום רבי שמעון שזורי פול המצרי שזרעו כזרע מקצתו השריש לפני ראש השנה אין תורמין ומעשרין מזה מזה על זה לפי שאין תורמין ומעשרין לא מן החדש על הישן ולא מן הישן על החדש**
>
> Isn't it taught in the Baraita: Rabbi Yosei ben Kei-far said in the name of Rabbi Shimon Shezuri: If the cowpea plant was planted for seed, not to be eaten as a vegetable but for the seeds, and some took root al-ready before Rosh HaShana, while some took root only af-ter Rosh HaShana, one may not set aside teruma or tithes

> from this for that, as one may not set aside teruma or tithes from the new crop for the old or from the old crop for the new. [47]

Here, the Gemara quotes the Baraita verbatim to bring a challenge to a previous claim. While the surrounding text is in Aramaic, this whole quote is in Hebrew.

While the content of these Tannaitic works is beyond the scope of this paper, it is important to note that these texts were composed using a very similar dialect of Hebrew as that in the Mishnah. Therefore, while the main text of the Mishnah is clearly marked in all modern printings of the Talmud, any Tannaitic works quoted within the Gemara are not cited.

The Mishnah and the other Tannaitic works in the Talmud are written in Tannaitic Hebrew (to be referred to simply as Hebrew). In contrast, the Talmud was written hundreds of years later in Jewish Babylonian Aramaic (to be referred to simply as Aramaic). Additionally, modern printings of Talmud usually do not include much punctuation. The main exception is a colon, which was introduced in the Vilna printing of the Talmud in order to separate topic shifts. However, these do not appear often enough to be of use for parsing the text.

Adding to the difficulty, the Talmud is written very tersely, often leading to later commentators arguing as to the precise meaning of the text. Some argue [36] that the Talmud was written this way in order to prevent non-learned people from reading it and misinterpreting it. As a result of the inherent ambiguity in parts of the text, there is currently no standard way to break up the Talmud. The breakup used in this paper follows that of Koren Publishers [44].

Chapter 5

Mishnaic Hebrew and Babylonian Aramaic

5.1 Overview

Hebrew is the language used in much of the Bible and the Mishnah, and it is spoken nowadays in Israel. For a large period of history, Hebrew ceased being a spoken language altogether due to the strong influence of other languages on the inhabitants of Israel [41]. The effects of foreign languages can be especially seen in Mishnaic Hebrew. After Cyrus the Great allowed the Jews to return to Israel from the Babylonian captivity, Aramaic began to be spoken in Israel along with Hebrew. Therefore, Mishnaic Hebrew, the dialect spoken at that time, was influenced heavily by Aramaic [9].

When designing NLP systems to handle Mishnaic Hebrew and Aramaic, there are three aspects which need attention [54]. These are lack of vocalization in the text, the complex morphology of words and spelling variants. The main discussion of this

section will focus on Hebrew, although much of it is also relevant for Aramaic due to the similarities in the languages.

5.2 Lack of Vocalization

In NLP, ambiguity of any kind can make parsing a phrase difficult. One type of ambiguity stems from homographs, which are words with the same spelling but different pronunciations. These are relatively rare in English, although some examples include 'bass', either referring to an instrument or a type of fish, and 'lead', a type of metal or a verb meaning to go in front of someone. Hebrew and Aramaic, however, have many more homographs. The reason for this is that nekudot (Hebrew vowels) are generally excluded from written texts, especially ancient ones. However, these vowels are critical for distinguishing different conjugations of the same root as well as completely different words. In modern Hebrew, the letters vav (ו) and yud (י) tend to be added to words in order to make the vocalization of a word clearer; however, this convention is much less frequent in ancient texts. We will discuss this convention more closely in section 5.4.

An example which demonstrates this is the word שמנה, which has at least a dozen possible vocalizations. A few of the possibilities are sh'mena (fat; feminine adjective), shimna (her oil; noun), shamna (she became fatter; verb), she-mana (that she counted; verb), shmone (eight; number) or shamenna (that her manna; noun). Each of these vocalizations leads to a very different meaning and often POS. It is estimated that over half of the words in modern Hebrew have homographs, with the average being around 2 homographs per spelling [32]. Although this analysis wasn't performed on ancient Hebrew, since the grammars of modern and ancient Hebrew are similar, this figure is still relevant.

5.3 Complex Morphology

Part of the confusion in pronunciation stems from the fact that many prepositions and conjunctions can be prefixes to words. Seven out of the 22 letters of the Hebrew alphabet can be prefixes (the acronym to remember them is כלב ומשה) which can make it difficult to determine whether the first letter of a word is a prefix or part of the root. Additionally, these prefixes often can be combined, and sometimes this even changes the meaning of the prefix. For example, the prefix כ generally means 'similar to'. The prefix ש means 'that'. When combined though (כש) it means 'when' as in the word כשאמר, meaning 'when he said'. It is even possible, although somewhat contrived, to have five prefixes, as in וכשמהבית, meaning 'and when from the house'. Similar to English, Hebrew has possessive suffixes. Unlike English, these suffixes include the person, referred to as pronominal suffixes, as in שולחנו, meaning 'his table', versus שולחנה, meaning 'her table'. Both prefixes and suffixes make determining the root of the word much more challenging in Hebrew than in English.

Verb conjugation is much more complex than in English, although conjugation often follows strict patterns. The vast majority of verbs have three-letter roots, and those with four-letter roots tend to be foreign words. Verbs are conjugated into seven binyanim, or constructions, which determine the general intention of the verb. These binyanim indicate whether a verb is passive, active or reflexive and whether it is causative, intensive or simple. While these definitions generally hold, there are many roots whose conjugation don't fit their binyan. An example is החוויר, meaning 'to turn pale', which is passive, although it is conjugated in a binyan which is generally active. Aramaic also has seven binyanim, corresponding to those in Hebrew, but the conjugations are different.

Noun conjugation is fairly simple. Nouns can be classified as being either absolute, construct or determined [26]. Absolute means the noun is in its lexical form, either singular or plural. For example, **ספר** or **ספרים**, meaning 'book' or 'books', are absolute. Construct refers to a noun being possesive or otherwise belonging to something. For example in the phrase **תזת נח**, meaning Noah's thesis, **תזת** is in construct form. A noun is determined if it has a determiner prefix (**ה**, meaning 'the') or it is a proper noun.

5.4 Spelling Variants

A final difficulty in parsing Hebrew is spelling variants. As mentioned above, certain letters can added to a word in order to compensate for the lack of nekudot, primarily yud (**י**) and vuv (**ו**). These letters act similar to vowels in English. Words written without vowels can be classified into two categories; ktiv male and ktiv haser. These mean, respectively, including vowel letters to replace nikud, and not including them. An example of the difference is **דבר** versus **דיבר**. Ktiv haser is very common in the Torah, making it extremely difficult to read without nekudot. The Academy of the Hebrew Language has made a convention for writing words in ktiv male [22]. However, no standard existed for adding these characters in ancient texts. Therefore, otherwise identical words might not match due to minor spelling variants.

Another source of spelling variants comes from transcription errors. These are especially prevalent in the Talmud, which underwent many printings, and to this day, scholars cannot agree on a critical version. Therefore, algorithms written to handle these ancient texts must be robust when dealing with words with minor spelling variants.

5.5 Aramaic

In many ways, Aramaic morphology is very similar to Hebrew [10]. For example, Aramaic shares most of the prefixes that exist in Hebrew. The prefixes which differ are ד, ל and א. The ד prefix replaces the ש in Hebrew. The prefix ל in Aramaic has two meanings. It can make a verb infinitive, like in Hebrew. Additionally, it can mark an object as definite, similar to the word את in Hebrew. Furthermore, the prefix א which means 'on' exists only in Aramaic.

Both Hebrew and Aramaic share the same consonants (letters) and vowels (nikud). However, their are consonants in one language which consistently are swapped with other consonants in the other language. For example, often the letter ז in Hebrew is swapped for a ד in Aramaic. This is demonstrated in the Hebrew word זהב meaning 'gold', versus the Aramaic equivalent, דהב.

Aramaic verbs are also based on conjugating roots . Talmudic Aramaic has 6 binyanim although other dialects contain more. [10], considering only the vocabulary in the Talmud, counts 6. However [27], considering Aramaic grammar as a whole, counts 8. Notably, many conjugations for verbs don't exist in the Talmud, and therefore grammarians need to predict their proper conjugation where it is not explicit.

Chapter 6

Related Work

6.1 Overview

While research into POS tagging for Talmud is lacking, this paper does build on the work of others in two key areas. We have divided our task into language identification and POS tagging. Specifically, we require word-level language identification, since Talmud is a mixture of Hebrew and Aramaic. For POS tagging, we are dealing with the issue of tagging a language with relatively little research in the NLP community.

6.2 Word-Level Language Identification

Language identification is often done on a document level. Document-level algorithms generally use bag of N-grams [6] which has proven very successful. [4] argues that language identification on a document level is largely a solved problem. However, document-level algorithms perform poorly when applied on a word-level in bilingual documents, as Nguyen shows in [40].

Nguyen deals with the problem of tagging each word in a bilingual document with its language. His dataset comes from a forum in the Netherlands used by Turkish-Dutch speakers. Posts on this forum are often bilingual due to the nature of the audience. He manually tagged words in over 2,000 posts as to whether they were Turkish or Dutch. Among his expirements, he tried using dictionary lookups, language models, and conditional random fields (CRFs). The language model was created using character n-grams. He found that language models did better than dictionaries, but that CRFs did the best, because they include context. He achieved 97% accuracy using CRFs.

In [7], Chittaranjan et al. deal with a similar problem, called code-switching. This is the problem of tagging words in a document based on the linguistic syntax and phonology of the words. An example discussed by Chittaranjan are tweets by people fluent in both English and Nepalese which borrow terms and syntaxt from each language. Each word is tagged with one of six tags, *lang*1, *lang*2, *mixed*, *ne*, *ambiguous* or *other*. Each document is a mixture of two languages. *lang*1 and *lang*2 refer to each of these languages, respectively, *mixed* indicates that the word contains morphemes from both, *ne* is a named entity, *ambiguous* is a tag which cannot be decided given the context and *other* is for punctuation and other characters. They trained their system for the following language pairs: English-Spanish (En-Es), English-Nepali (En-Ne), English-Mandarin (En-Cn), Standard Arabic-Dialectal Arabic (Ar-Ar).

The authors found that using a CRF classifier with n-gram features was helpful. Using the features from previous words actually decreased accuracy, but just looking at previous tokens helped. Overall, they achieved accuracies around 90% on their test sets for the various language pairs. They found that the Ar-Ar pair did not benefit

as much from n-gram features probably because these features are similar for both standard and dialectal Arabic.

From a linguistic perspective, it is unclear whether the Talmud includes code-switching or whether it is just a mixture of Aramaic and Hebrew. Research in this field is scarce. The Academy of Hebrew Language [23] has done extensive research into which words have Hebrew origins in Talmud, but unfortunately we were not able to obtain their dataset for our research. Since we didn't have a dataset for Talmud tagged by language, we were unable to implement the final solutions of either [40] or [7] which both require context features. However, we were able to use insights from their research. [40] shows that language models are more powerful than dictionaries, even without context. We found this to be true for Talmud. [7] shows that n-gram features were helpful for the majority of their language pairs except Ar-Ar. We found that their results for Ar-Ar language pair were not indicative of our results for Aramaic-Hebrew although both language pairs are semitic. We used a BiLSTM on the characters of individual words as our language model, which is similar to an n-gram language model [19]. We included a third tag, ambiguous, also used in [7]. This tag accounted for the many Hebrew words which are borrowed in Aramaic.

6.3 POS tagging

In [1], Adler addresses the problem of morphologically analyzing unknown words in Modern Hebrew. This is the task of giving a distribution of possible tags for a given word. His motivation for this is to use the analysis as input to a standard HMM POS tagger. As noted by [21] simply using a rule-based approach to generate all possible tags for a word gives far too many options to be useful for an HMM POS tagger. Instead, Adler compared various features for a maximum entropy model.

These features included:

- Letter - Generate each possible word by stripping off possible prefixes; for each word, make a feature vector made up of 1, 2 and 3 letter n-grams from the word
- Pattern - A list of 40 binary features corresponding to patterns of conjugation in Hebrew; a feature is 1 if the given word falls into that pattern
- WordContext - A feature vector including the previous, current and next word
- TagContext - A feature vector including the previous, current and next tag

They found that the combination of features which achieved the highest POS tagging accuracy was Pattern-Letter, with 78.5%. However, WordContext-Letter performed almost equally well, at 78.2%. This is significant, because WordContext is based only on the input text and Letter is based on morphological analysis of known words. This implies that in Hebrew the tags for unknown words can be inferred from just this information.

In [53], Wang experiments with using a BiLSTM for POS tagging in English. He first trains embedding vectors for the words in his corpus using an unsupervised approach. For each sentence, he randomly replaces some words with a random word from the corpus. The tag for the words in the sentence is then INCORRECT if it has been replaced and CORRECT if it hasn't. The final vectors in the embedding matrix after training are then used for POS tagging. The features for the POS tagging BiLSTM are:

- Capital Vector: Embedding vector with three possible values: All Capital, Leading Captial, No Capital

- Word Embedding: Embedding vector for the current word
- Suffix Ngram: One-vector representing the bigram of the last two characters in a word

Wang compares different sizes of datasets to train his word embeddings in the pre-processing stage. A dataset of 10 million words did not affect the performance as compared to using random word embeddings. However, including all 530 million words in the dataset showed a large difference in performance. He was able to achieve state-of-the-art accuracy of 97.40%, better than the previous highest accuracy of 97.36%. His conclusion is that a BiLSTM is strongly suited for POS tagging and that it is not necessary to include complex morphological features to achieve high performance.

Goldberg in [19] discusses various ways to train embedding vectors. The most obvious approach is to use an embedding vector per word in the corpus. However, this has two disadvantages. This requires a very large dataset (as shown by [53]) to train and it does not generalize well for words that do not appear in the dataset. Another approach is to use embedding vectors for characters. Then, the vector for a word is simply the concatenation of the outputs corresponding to the first and last letters of the word as output by a character BiLSTM. In a language where the syntactic function of a word is highly correlated with its character patterns, using character vectors instead of word vectors can be a better strategy.

Based on [53], we have used a BiLSTM for Talmud POS tagging, considering that its performance has surpassed that of HMM POS taggers. We note that our problem of POS tagging in the Talmud is harder than a conventional case because the Talmud has no punctuation and there are Hebrew words mixed with the Aramaic.

We therefore choose a BiLSTM because they can learn dependencies on arbitrarily long sequences, as opposed to HMMs which are limited. However, unlike Wang, we do not have the convenience of a large untagged dataset. The Talmud contains only about 1.8M words, less than the 10M words Wang showed were insufficient to learn proper word embeddings. Additionally, Aramaic words contain much more variance than English words since prepositions, conjuntions and pronouns are appended to the word. This implies that Aramaic would require even more unsupervised training than English for word embeddings. However, as [1] shows, Hebrew, and presumably Aramaic considering the word structures are similar, can be effectively modeled using a character n-gram language model. Goldberg suggests using character instead of word embeddings for languages where syntax can be deduced from character patterns. Therefore, we chose to use character embeddings to model the words in Talmud.

Chapter 7

Data and Resources

7.1 Data

There were two main sources of data used for our research. The first is a partially tagged dataset of the Talmud from the Comprehensive Aramaic Lexicon (CAL) [27]. The second is the full text of Mishnah and Talmud from Sefaria [34].

7.1.1 CAL Dataset

CAL is a project started in 1986 to create a comprehensive dictionary for every dialect of Aramaic. According to [27], no project of this scope has ever been made. Previous Aramaic dictionaries focused either on a certain dialect or on the language in a specific text. For example, the Jastrow Aramaic dictionary covers language found in the Midrash, the Targum (a corpus of Aramaic translations of the Bible), and the Talmud. However, this dictionary also includes many Hebrew words found in Talmud. In contrast, CAL tries to only include Aramaic words in its dictionary.

The CAL dataset is divided into two main portions, the dictionary and tagged texts. For the purpose of this thesis, the tagged texts were used. Specifically, we received 7 out of the 37 tractates of the Babylonian Talmud in order to train on. These were tractates Berakhot, Shabbat, Eruvin, Pesachim, Bava Kamma, Bava Metzia and Bava Batra. However, because CAL only includes Aramaic in their dataset, these texts were fragmentary. In places where CAL considered a word to be Hebrew, it is completely omitted from the data.

Each word in the dataset contains the following information:

- Location information - this includes the book id, page number, line number and word number; these follow the Vilna edition
- Head word
- Part-of-speech
- Written word as it appears in text
- Prefixes and POS for prefixes if applicable; this data was not used for this project

Table 7.1 shows the POS included in the dataset. As noted in section 5.3, CAL includes 8 verb binyanim although only 6 are known to exist in the Talmud. The final tagset used for training was extracted from the unique tags in the dataset, and therefore all unused tags were removed. These were V07, V10 and V11.

As outlined in [28], there are several problems with the CAL dataset. CAL intended to only tag Aramaic words in the Talmud. However, they missed many Aramaic

POS tag	Meaning	POS tag	Meaning
N01	Noun singular absolute or construct	A01	Adjective singular absolute or construct
N02	Noun singular determined	A02	Adjective singular determined
N03	Noun plural absolute	A03	Adjective plural absolute
N04	Noun plural construct	A04	Adjective plural construct
N05	Noun plural determined	A05	Adjective plural determined
V01	Verb binyan peal	n01	Number singular absolute or construct
V02	Verb binyan pael	n02	Number singular determined
V03	Verb binyan haphel	n03	Number plural absolute
V04	Verb binyan ethpeel	n04	Number plural construct
V05	Verb binyan ethpaal	n05	Number plural determined
V06	Verb binyan ettaphal	p01	Preposition independent
V07	Verb binyan payael	p02	Preposition with pronominal suffix
V08	Verb reduplicated	p03	Preposition proclitic
V09	Verb quadriliteral	GN	Geographical name
V10	Verb binyan ethpaiel	c	Conjunction
V11	Verb reflexive reduplicate	a	Adverb
V12	Verb reflexive quadriliteral	I	Interjection
PN	Proper name		

Table 7.1 List of POS in the CAL dataset

words, especially those found in Hebrew contexts such as a quote from the Mishnah. Finally, the dataset contains many typographic errors. These errors include:

- Misaligned headwords - the headwords and written words are not aligned in a section
- Tagging base words with prefix POS - generally, when a word has a prefix, both the POS of the prefix and base word are tagged; however, frequently only the POS of the prefix was tagged
- Incosistent tagging schemes - sometimes, a word would be repeated twice, the first time with the base word's POS and the second time with the prefix's; sometimes these were included on one line.

These issues are very problematic for training a POS tagger. Inconsistent tags for words makes it very difficult to learn the proper usage of a word. We manually fixed about 15,000 errors which we found in the dataset.

7.1.2 Sefaria Dataset

Sefaria is an open-source project which aims to upload the entire corpus of Jewish literature online and with creative-commons licenses. Their database contains over 90M words, including the Bible, the Mishnah and the Talmud as well as many other texts. We used two texts in this thesis, the Mishnah and the Talmud. Although the Talmud contains all of Mishnah, Sefaria has a text which contains only the Mishnah. We used this to train our language identification system, described in Chapter 8. Their version of the Talmud comes from Koren [44]. What is notable about this edition is that it is segmented into semantically significant sections. While the Koren edition is based on the Vilna edition, the latter contains no punctuation, which makes

parsing it difficult. We experimented using Sefaria's segmentation to improve POS tagging accuracy. Also notable is that Koren opens up abbreviations found in the Vilna edition of the Talmud. We only used the 7 tractates which we had from CAL.

7.1.3 Final Dataset

The CAL dataset only includes Aramaic and not Hebrew. Also, it often skips Aramaic words which appear in Hebrew contexts. This means the text is extremely fragmentary making it unusable for training a POS tagger, which requires context. In order to use the dataset, we first had to align it to a the full text of the Talmud. We chose the Koren edition of Talmud available on Sefaria as our edition of Talmud.

This alignment was difficult for three reasons. First, as discussed in chapter 5, ancient Hebrew and Aramaic have no standard spelling. Two otherwise identical words can be spelled differently. Second, CAL does not always use the Vilna edition of Talmud, which Koren is based on. This introduces words which appear in one edition but not the other. Third, CAL often uses abbreviations found in the Vilna edition while Koren always opens these.

We wrote a scipt which deals with these issues. It uses a weighted Levenshtein distance [49] to compare word similarity. This is similar to Levenshtein distance, which calculates the edit distance between two strings, except individual edits are weighted, depending on the character being edited. The weights we use are based on the frequencies of letters in the Talmud. For example, י is the most common letter and therefore contributes the smallest distance when calculating the overall levenshtein distance between two words. This metric is useful when comparing minor spelling variations since י and ו are the most common variances in spelling and also

the most frequent letters. The script does multiple passes over the same page of the Talmud, trying to find the optimal alignment for that page. Finally, it includes an abbreviation matching algorithm.

Our script matched 112,426 (76.0%) of the words in the CAL dataset. While this number is relatively low, we note that the version differences account for many words which don't appear in the Koren edition. 18.8% of the Koren text matched words in CAL. The total size of the Koren text used is 599,437.

7.2 Resources

7.2.1 Dynet

We used Dynet [39] as our neural network library. It is a library written in C++ with bindings to Python. It has a simple implementation of a BiLSTM and many APIs useful for NLP. It was developed by Carnegie Mellon University along with many other researchers.

7.2.2 Sefaria Codebase

We used the Sefaria codebase [35] which has a data model to query their dataset. We used it to a access pages of the Talmud and paragraphs of the Mishnah.

Chapter 8

Methodology and System Design

8.1 Overview

In this chapter, we outline the system design used in the Part of Speech Handling for Aramaic in the Talmud (PSHAT) system. Below is a summary of the pipeline from beginning to end:

1. Read in data from combined CAL / Sefaria dataset (see section 7.1.3)
2. Run data through language identification system; tag words as either Aramaic or Hebrew
3. Run language-tagged-data through POS BiLSTM; this only predicts POS for words which have the 'Aramaic' tag

8.2 Word-Level Language Identification

There are two reasons we include a language identification step before POS tagging. The first is practical; the CAL dataset only includes tags for Aramaic words. Any

model trained only on Aramaic will likely perform poorly in a full text of the Talmud, which has both Hebrew and Aramaic. The second is that while Aramaic and Hebrew grammar are related, there are significant differences. Trying to train a system to tag both languages at the same time would likely produce poor results. It is important to note that we could not simply say that all the words included in the CAL dataset are Aramaic and all excluded are not, because of the inconsistencies in CAL's tagging as mentioned in section 7.1.1. There are many Aramaic words excluded from their dataset.

Under the assumption that the words in CAL are a good representation of Aramaic in the Talmud, and that the words in the Mishnah are represenative of the Hebrew in the Talmud, we created a training dataset. Unlike [7] and [40], we did not have a tagged dataset for which words are Aramaic and which are Hebrew in the Talmud. Instead, we trained a classifier to classify words out of context. It used three tags, Hebrew, Aramaic and ambiguous. We then applied a simple heuristic to disambiguate the words with the 'ambiguous' tag.

The training set included 114,755 Aramaic words from CAL and 118,782 Hebrew words from the Mishnah. A word was considered ambiguous if it appeared in both CAL and the Mishnah more than 10 times. There were 9,267 ambiguous words. We included an ambiguous tag because, given a word out of context, it is not always possible to decide definitely what language it originates from. Additionally, as discussed in [7], when languages are used interchangably in similar contexts, language boundaries become blurred. For example, the word שחרית ('morning') is Hebrew. However, the Talmud might use this word with an Aramaic prefix such as in the case of דשחרית ('of morning'). It is unclear whether this word should be classified as either Aramaic or Hebrew. A more common case where language is ambiguous is with stop

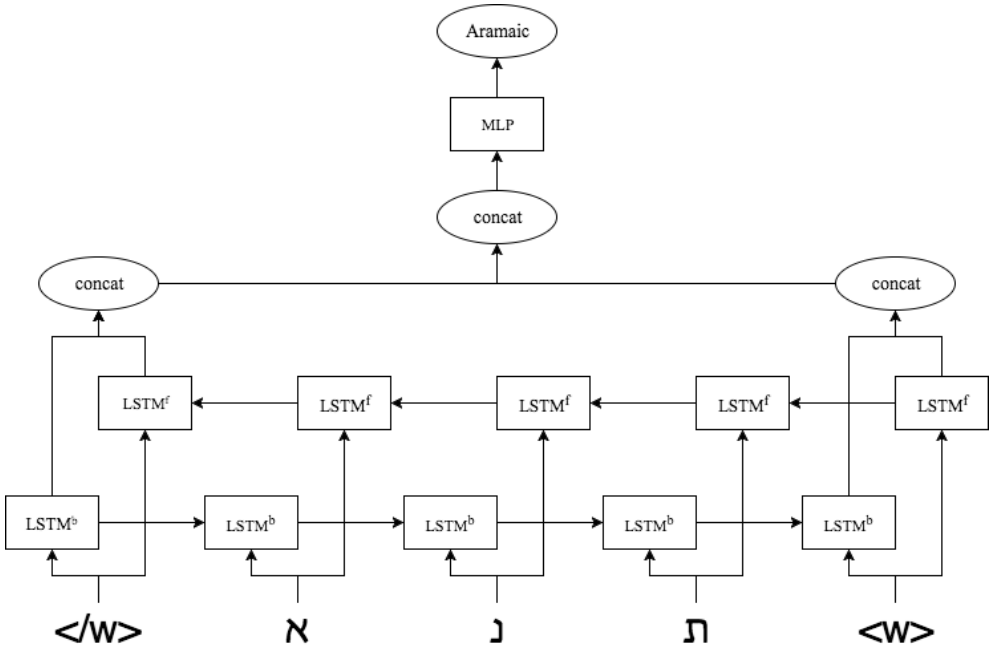

Figure 8.1 An example input to our language identification system. The word תנא is input to the BiLSTM. The output of the BiLSTM is fed to an MLP which predicts the language as Aramaic

words. Words such as לא (‘no’) and כל (‘all’) are used frequently in both Aramaic and Hebrew contexts.

We chose a BiLSTM classifier to classify words as Hebrew, Aramaic or ambiguous, as shown in Figure 8.1. The input to the BiLSTM is the embedding vectors for the characters of a word, padded with start and end word tags. Note that Aramaic and Hebrew are read from right-to-left. Therefore, the backwards LSTM, shown above the input, is reading left-to-right. The first and last outputs from the LSTMs were concatenated by appending their elements to each other. This vector is fed into an MLP (feedforward network) which predicts the langauge.

[7] shows that n-gram language models are effective when dealing with code-switching. We chose a BiLSTM language model based on characters, because [19] explains how

these are similar to n-gram models, but are more flexible. This is because the model doesn't need to learn explicit 3-grams or 2-grams, but rather can learn the optimal character patterns to minimize the loss function.

After tagging the words in our dataset with the BiLSTM language identifier, we applied a heuristic to remove the words with the 'ambiguous' tag. We applied the decision rule indicated in Equation 8.1 to determine each tag.

$$isAr = (mean(C_{ar,i-1}, C_{ar,i+1}) + C_{ar,i}) > (mean(C_{he,i-1}, C_{he,i+1}) + C_{he,i}) \quad (8.1)$$

If $isAr$ is true, the word at position i is Aramaic; otherwise, it is Hebrew. $C_{ar,i}$ is the confidence, as output by the MLP in the language identifier, that the word at position i is Aramaic. $C_{he,i}$ is the confidence that the word is Hebrew. This decision rule checks whether the average of the confidences of the surrounding words plus the confidence of the current word is higher for Aramaic or Hebrew.

8.3 POS Tagging

Our full POS tagger system is shown in Figure 8.2. The example input only includes two words, but we actually input a page of the Talmud at a time, with beginning and end sequence tags before and after, respectively. These tags are treated as seperate characters. SP represents a space. The characters are first input into the forward and backward LSTMs as described in the previous section. The outputs of the BiLSTM before and after each word as well as the embedding vector of the previous POS are concatenated. At this point we check the language tag of the current word, as determined by the language identifier. If it is Hebrew, we give it the special HE POS

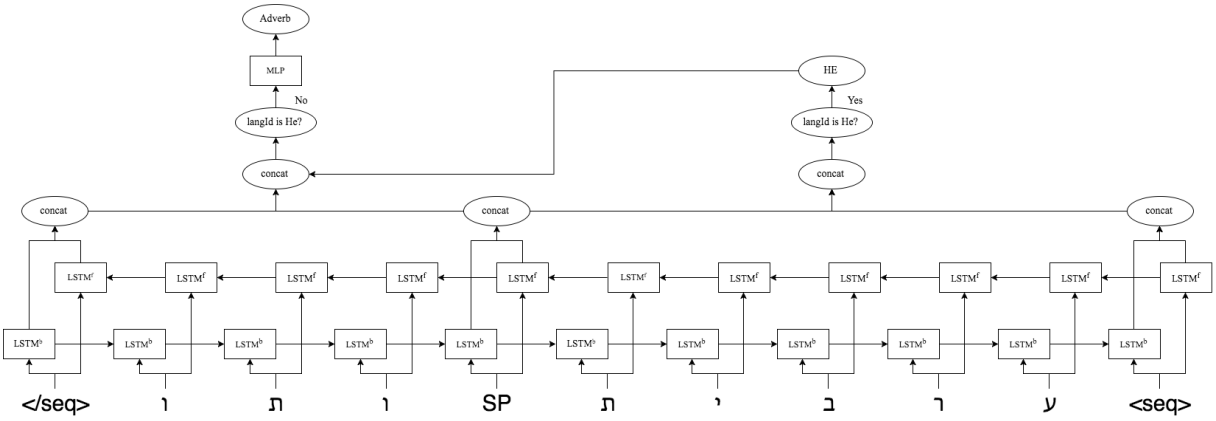

Figure 8.2 An example input to the POS tagger. This passage contains two words; the first is Hebrew and the second is Aramaic

tag. Otherwise we pass the concatenated vector to an MLP which predicts the POS. This process continues for all words on the page.

A dictionary is used to prune the possible POS for a given word. The keys in the dictionary are words in our corpus. The value for a word is a list of the POS used for this word in the training set. After the MLP output is generated, the list of possible tags is retrieved based on the current word and the output is trimmed to only include these tags. The maximum value from this set is the final POS tag for that word.

We trained our model in a unique way, based on the nature of our data. Generally when training a BiLSTM model, the loss is accumulated over the entire sequence (in our case, a page of the Talmud) and then the backpropogation algorithm is run. However, our sequence contains two types of words for which the loss should not be calculated. The first is Hebrew words; our language identifier system already tagged these and the POS tagger does not need to make a decision for these. The second is Aramaic words not included in the CAL dataset. While we are interested in the output of our system for these words, we have no training data to calculate a loss. Therefore, our system does not include these types of words in the loss calculation.

Chapter 9

Experimentation and Results

9.1 Overview

For all our experiments, we shuffled our data and split it into 80% training, 20% testing. Between epochs of training, we shuffled the training data. We used the same random seed for all experiments in order to better notice changes in performance when modifying parameters. We used a momentum stochastic gradient descent trainer for our neural network optimizer [45].

9.2 Initial Experiments

Using the BiLSTM architecture described in section 8.3, we trained our POS tagger initially without distinguishing between Hebrew and Aramaic words. All words in our dataset without POS tags were assigned a special unknown tag. We used an embedding dimension of 50, a 2-layer BiLSTM, and a 3-layer MLP with hidden layers of dimension 100.

Results for this experiment led to overfitting; the training accuracy was 100% while the validation accuracy was 63%. We hypothesized this experiment overfit because the training data had many conflicting tags for words. As mentioned in section 7.1.1, the CAL dataset does not include POS tags for all Aramaic words in the Talmud. Therefore, for certain Aramaic words, our dataset sometimes included a POS tag for the word, but other times the POS tag was unknown. The network could not generalize these patterns when running on validation data. To solve this issue, we only included words with POS tags in our loss calculation, as described in section 8.3.

9.3 Language Identification

By no longer training on the unknown tags in our dataset, our model had no opportunity to distinguish between Aramaic and Hebrew words. We therefore added in a seperate language identification stage. We used an embedding dimension of 20, a 2-layer BiLSTM, and a 2-layer MLP with a hidden layer of dimension 40. We achieved **93.2%** and **93.0%** for training and validation accuracy respectively.

9.4 POS Tagging

We first computed a reasonable baseline with which to compare our results. For our baseline, we split our dataset 80/20 for training and testing. Based on the training data, we made a dictionary mapping every word to its most likely POS tag. Running on the testing data, we chose the most likely POS assuming it existed in the dictionary, otherwise we chose V01, which was the most likely POS in the training set. This system achieved an accuracy of 54.4% and 47.2% on the training and testing sets respectively.

Beam Width	Training	Testing
1	95.10%	90.81%
3	96.80%	90.63%
5	97.12%	90.50%
10	98.05%	90.88%

Table 9.1 Beam search POS tagging accuracy

Using the BiLSTM described in section 9.2, we tested our POS tagger. We trainined using beam search and without beam search. We also varied the segmentation of the training data to see how this affected accuracy. We compared training on the standard Vilna page segmentation versus the Koren segmentation which roughly corresponds to phrases. As discussed in section 3.3, beam search performs best for training neural networks when combined with early update. We found that using early update for training on page segmentation led to overfitting of the model. This was mostly likely due to the fact that early update will stop training on a segment early if the correct POS for a word falls out of the beam. This means that it will be very unlikely that the model will completely train on long segments. We therefore only show results for beam search trained on Koren's segmentation.

When reporting accuracy for POS tagging, it is important to note that it is dependent on the accuracy of the language identifier. If the language identifier tags an Aramaic word as Hebrew, the POS tagger will skip this word, effectively giving it the wrong POS. It is impossible to quantitatively take this error into account since we don't have a dataset for the Talmud tagged by language to compare with. Our POS tagging accuracies reflect performance only for words tagged as Aramaic. In general, this means the POS tagging accuracy is lower than it is reported.

Training Segmentation	Training	Testing
Page	97.42%	90.97%
Paragraph	94.86%	90.88%

Table 9.2 Page vs paragraph training POS tagging accuracy

9.5 Discussion

The results for modifying the beam width are shown in Table 9.1. We varied the beam width between 1 and 10. While we saw the training accuracy increase steadily, the testing accuracy remained about constant, indicating that the training increase was only due to better fitting of the model to the training set. According to these results, beam search is not helpful for learning POS in the Talmud. This may indicate that sentence structure in the Talmud is not very ambiguous and can often be tagged correctly using a greedy approach. However, this hypothesis needs to be researched further.

The results for different training segmentations are shown in Table 9.2. Note that we ran this experiment without beam search because beam search overfit when training on page segments, which tend to be very long. Segmentation is surprisingly not a factor for the accuracy of our system. This could imply that the BiLSTM used was able to learn the proper breakup of phrases without explicit segmentation. Our final implementation used the page segmentation without beam search since this system performed slightly better on training and testing. Our final accuracies for words tagged as Aramaic were **97.42%** and **90.97%** for training and testing respectively. We note that these accuracies outperform our baseline.

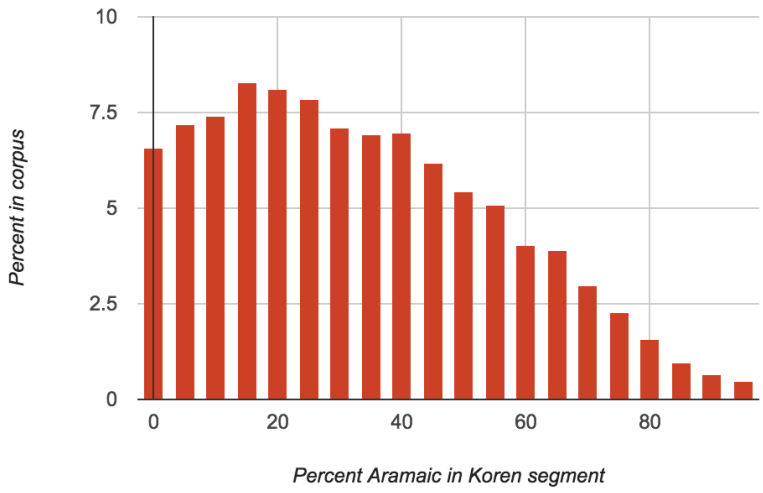

Figure 9.1 Relative frequency of segments by percent Aramaic

Of interest is the performance of the POS tagger in cases where there is a lot of Aramaic versus where the Aramaic is sparse and mixed with Hebrew. To evaluate this, we first created a histogram counting the number of Koren segments based on the percentage of Aramaic they contained, shown in Figure 9.1. The curve follows a rough Gaussian, with mean around 20% Aramaic. It is interesting to note that the majority of segments contain less than 50% Aramaic, and very few are completely Aramaic.

We then calculated the accuracy of the POS tagger by the percentage of Aramaic per segment. The results are shown in Figure 9.2. As can be seen, the accuracy of the tagger is relatively constant with respect to the percentage of Aramaic in the segment with the exception of segments with very few Aramaic words. This shows that our tagger is robust to language mixtures of Hebrew and Aramaic. This is a very interesting result considering POS tagging requires context in order to disambiguate the multiple POS tags possible for a given word. It is also interesting because of

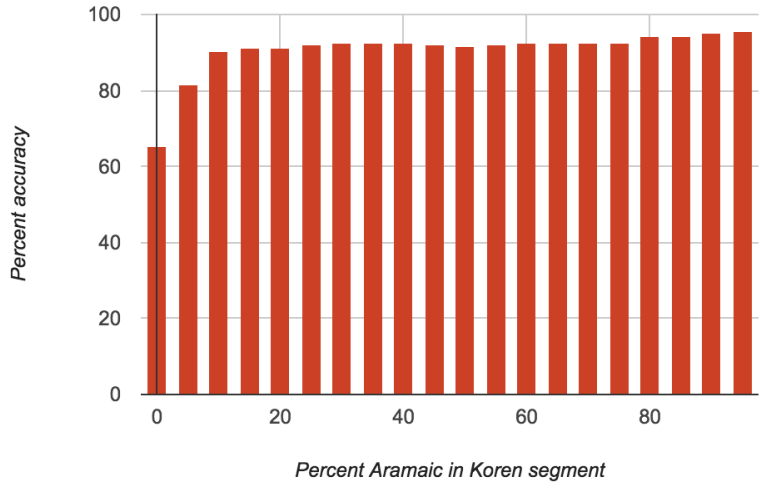

Figure 9.2 Accuracy of POS tagger by percent Aramaic

the high ambiguity in Hebrew as compared to English, discussed in Chapter 5. The BiLSTM is likely the reason the system can handle language mixing so well.

We also checked the performance of the system on unknown words. Unknown words are those which appear in the test set, but not in the training set. Generally, systems perform far worse on these words since they were not able to train on them. We hypothesized that our system would still perform reasonably well on unknown words. The reason for this is that we used a BiLSTM trained on characters, which has the ability to learn inflections and other syntactic patterns in a language.

We tested our hypothesis by comparing our system against a modified baseline. This new baseline first predicted the most likely POS for a word, assuming it had seen it before. Otherwise, it chose the most likely POS for the head word, assuming it had seen that. Otherwise it guess V01, the most common POS in our training set. This baseline makes the assumption that the head word of word is known in order

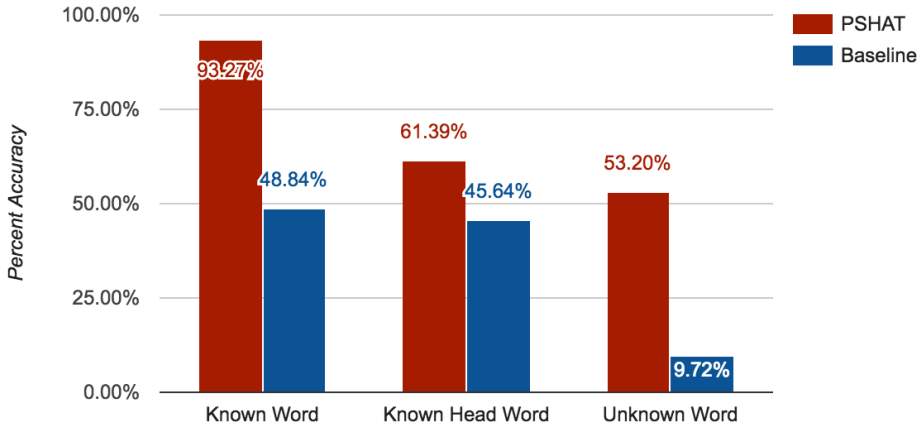

Figure 9.3 Comparison of accuracy on both known and unknown words

to compare it against our system. We divided the words in the test set into three categories; namely, known words, known head words and unknown words. Known words are words that appear both in the training and test sets. Known head words are words whose head words appear in the training set, but not the word itself. Unknown words only appear in the test set.

Figure 9.3 shows the results of this experiment. As expected, our system far outperformed our baseline for known words. Very interestingly, it also outperformed the baseline for known head words and unknown words. This implies that our system was able to infer syntactic information from the characters of the word itself, without having to see the word explicitly. Note that the performance for known head words is better than that of unknown words, which suggests that the system was able to take advantage of the head word to more accurately predict the POS.

Chapter 10

Conclusion

POS tagging involves labeling words in a text with their correct syntactic category. It is difficult because often there exist multiple possible categories for a single word. Using context can help disambiguate these possibilities. Previous solutions have used rule-based methods and HMMs to tag texts with POS. Recently, neural networks have been applied to this task, with LSTMs achieving state-of-the-art accuracy.

However, these approaches have focused on modern texts. Little work has been devoted to tagging ancient texts, which are often more ambiguous and difficult to parse. We focused on POS tagging in the Talmud which posed two unique difficulties; it does not include any punctuation, and it contains a mixture of Aramaic and Hebrew. These challenges made it impossible to apply current state-of-the-art techniques.

Our solution used a two-step approach. First we trained a language identifier BiLSTM to distinguish between Hebrew and Aramaic in the Talmud. We then trained a BiLSTM POS tagger to tag only the Aramaic words. We achieved 93.00% and 90.97% accuracy for our language identifier and POS tagging, respectively. We also validated that our model is robust when dealing with varying mixtures of Aramaic

and Hebrew. Also, the fact that our system performs similarly training on pages or paragraphs implies that our system is robust to segmentation. Finally, we showed that we far outperformed our baseline for unknown words. We also experimented using beam search to improve our results for POS accuracy, since it allows the tagger to use global context. However, it did not increase our accuracy .

Because we built our system with few assumptions that were specific to the Talmud, we believe this work should be valuable for other texts with language mixtures. The BiLSTMs used are very flexible and should be easily applied to other languages besides Hebrew and Aramaic.

There are a few areas of this project that could be improved with future work. The first would involve creating a dataset for the Talmud which has words tagged by language. This might be as simple as tagging each word as Hebrew or Aramaic. However, further research might reveal that the Talmud employs code switching. If this is true, the tags would need to include more detailed information about the language properties of each word. Another area of improvement would be including a named-entity recognition stage. This stage would likely improve POS tagging accuracy, since it would give the tagger more detailed information. A third improvement would be utilizing the prefix tags in the CAL dataset. We did not use these because we found them to be unreliably tagged. However, if this data were improved, it could be used as additional features for our system. Finally, our dataset only included 7 out of the 37 tractates of the Talmud. If CAL releases the rest of the dataset, this could be used to further generalize our model.

Bibliography

[1] Meni Adler et al. "Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morpholological Analysis". In: *In Proceedings of ACL-08*. 2008.

[2] Apoorv Agarwal et al. "Sentiment Analysis of Twitter Data". In: *Proceedings of the Workshop on Languages in Social Media*. LSM '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 30–38. ISBN: 978-1-932432-96-1. URL: http://dl.acm.org/citation.cfm?id=2021109.2021114.

[3] ES Atwell. "Development of tag sets for part-of-speech tagging". In: (2008).

[4] Shane Bergsma et al. "Language identification for creating language-specific twitter collections". In: *Proceedings of the second workshop on language in social media*. Association for Computational Linguistics. 2012, pp. 65–74.

[5] Thorsten Brants. "TnT: A Statistical Part-of-speech Tagger". In: *Proceedings of the Sixth Conference on Applied Natural Language Processing*. ANLC '00. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 224–231. DOI: 10.3115/974147.974178. URL: http://dx.doi.org/10.3115/974147.974178.

[6] William B Cavnar, John M Trenkle, et al. "N-gram-based text categorization". In: *Ann Arbor MI* 48113.2 (1994), pp. 161–175.

[7] Gokul Chittaranjan et al. "Word-level language identification using crf: Code-switching shared task report of msr india system". In: *Proceedings of The First Workshop on Computational Approaches to Code Switching*. 2014, pp. 73–79. URL: http://www.anthology.aclweb.org/W/W14/W14-39.pdf#page=81.

[8] Michael Collins and Brian Roark. "Incremental parsing with the perceptron algorithm". In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2004, p. 111.

[9] Miguel Perez Fernandez. *An Introductory Grammar of Rabbinic Hebrew*. Koninklijke Brill, 1997.

[10] Rabbi Yithak Frank. *Grammar for Gemara and Targum Onkeles*. 2nd. Maggid, 2016.

[11] David Furcy and Sven Koenig. "Limited discrepancy beam search". In: *IJCAI*. 2005, pp. 125–131.

[12] Roger Garside. *The CLAWS word-tagging system In The Computational Analysis of English: : A Corpus-Based Approach*. Longman, 1987.

[13] Kevin Gimpel et al. "Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 42–47. ISBN: 978-1-932432-88-6. URL: http://dl.acm.org/citation.cfm?id=2002736.2002747.

[14] Yoav Goldberg. "A primer on neural network models for natural language processing". In: *Journal of Artificial Intelligence Research* 57 (2016), p. 356.

[15] Yoav Goldberg. "A primer on neural network models for natural language processing". In: *Journal of Artificial Intelligence Research* 57 (2016), p. 364.

[16] Yoav Goldberg. "A primer on neural network models for natural language processing". In: *Journal of Artificial Intelligence Research* 57 (2016), p. 389.

[17] Yoav Goldberg. "A primer on neural network models for natural language processing". In: *Journal of Artificial Intelligence Research* 57 (2016), p. 399.

[18] Yoav Goldberg. "A primer on neural network models for natural language processing". In: *Journal of Artificial Intelligence Research* 57 (2016), p. 396.

[19] Yoav Goldberg. "A primer on neural network models for natural language processing". In: *Journal of Artificial Intelligence Research* 57 (2016), p. 369.

[20] Grayzel. *A History of the Jews*. Penguin Books, 1984.

[21] Nizar Habash and Owen Rambow. "MAGEAD: a morphological analyzer and generator for the Arabic dialects". In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 2006, pp. 681–688.

[22] Academy of Hebrew Language. *Haketiv Chaser Hanikud*. URL: hebrew-academy.org.il/topic/hahlatot/missingvocalizationspelling/.

[23] Academy of the Hebrew Language. *Tractate Berakhot*. URL: maagarim.hebrew-academy.org.il/Pages/PMain.aspx?mishibbur=80001.

[24] Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '04. New York, NY, USA: ACM, 2004, pp. 168–177. ISBN: 1-58113-888-1. DOI: 10.1145/1014052.1014073. URL: http://doi.acm.org/10.1145/1014052.1014073.

[25] ICAME. *Brown Corpus Manual*. URL: http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM.

[26] Blair Kasfeldt. *Biblical Hebrew Made Easy!* URL: http://biblicalhebrewmadeeasy.weebly.com/.

[27] Steve Kaufman. *The Comprehensive Aramaic Lexicon*. URL: http://cal1.cn.huc.edu/.

[28] Steve Kaufman. *What's the matter with Babylonian Aramaic?* URL: blog.huc.edu/cal/2016/03/whats-the-matter-with-babylonian-talmudic/.

[29] Eliyahu Kiperwasser and Yoav Goldberg. "Simple and accurate dependency parsing using bidirectional LSTM feature representations". In: *arXiv preprint arXiv:1603.04351* (2016), p. 318.

[30] Sheldon Klein and Robert F. Simmons. "A Computational Approach to Grammatical Coding of English Words". In: *J. ACM* 10.3 (July 1963), pp. 334–347. ISSN: 0004-5411. DOI: 10.1145/321172.321180. URL: http://doi.acm.org/10.1145/321172.321180.

[31] Philipp Koehn et al. "Moses: Open Source Toolkit for Statistical Machine Translation". In: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. ACL '07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 177–180. URL: http://dl.acm.org/citation.cfm?id=1557769.1557821.

[32] Moshe Levinger, Alon Itai, and Uzzi Ornan. "Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew". In: *Computational Linguistics* 21.3 (1995), pp. 383–404.

[33] Dr. Min Liu. *Parts of Speech*. URL: edb.utexas.edu/minliu/pbl/ESOL/help/libry/speech.htm.

[34] Brett Lockspieser. *Sefaria Project*. URL: https://sefaria.org.

[35] Brett Lockspieser. *Sefaria Project Github*. URL: https://github.com/Sefaria/Sefaria-Project.

[36] Maimonides. *Guide for the Perplexed, Introduction, Prefatory Remarks 6*. URL: http://www.sefaria.org/Guide_for_the_Perplexed.

[37] Andrew McCallum and Wei Li. "Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 188–191. DOI: 10.3115/1119176.1119206. URL: http://dx.doi.org/10.3115/1119176.1119206.

[38] MQL5. *Next price predictor using Neural Network - indicadores para MetaTrader 4*. URL: https://www.mql5.com/pt/code/9002.

[39] Graham Neubig et al. "DyNet: The Dynamic Neural Network Toolkit". In: *arXiv preprint arXiv:1701.03980* (2017).

[40] Dong Nguyen and A. Seza Dogruoz. "Word level language identification in online multilingual communication". In: *2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*. USA: Association for Computational Linguistics, 2013, pp. 857–862. URL: http://doc.utwente.nl/88555/.

[41] Nicholas Ostler. *Empires of the Word: A Language History of the World*. Harper Perennial, 2006, p. 80.

[42] University of Pennsylvania. *POS tags in Penn treebank*. URL: ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[43] Juan Antonio Perez-ortiz, Mikel L. Forcada, and Departament De Llenguatges I Sistemes. "Part-ofspeech tagging with recurrent neural networks". In: *In Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. 2001.

[44] Koren Publishers. *Koren Talmud*. URL: https://www.korenpub.com/.

[45] Sebastian Ruder. *Optimizing Gradient Descent*. URL: sebastianruder.com/optimizing-gradient-descent/index.html#momentum.

[46] Sefaria. *Tractate Berakhot*. URL: http://www.sefaria.org/Berakhot.2a.7.

[47] Sefaria. *Tractate Rosh Hashanah*. URL: http://www.sefaria.org/Rosh_Hashanah.13b.7.

[48] Eliezer Segal. *A Page from the Babylonian Talmud*. URL: https://people.ucalgary.ca/~elsegal/TalmudPage.html.

[49] Stanford. *Edit Distance*. URL: http://nlp.stanford.edu/IR-book/html/htmledition/edit-distance-1.html.

[50] *Talmud Yerushalmi*. B'rachot, Friedman's Oz ve-Hadar edition. Vol. vol 1. Oz ve-Hadar, 2010, Introduction, p. 17.

[51] Brown University. *POS tags in Brown Corpus*. URL: http://www.scs.leeds.ac.uk/amalgam/tagsets/brown.html.

[52] Peilu Wang et al. "Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network". In: *CoRR* abs/1510.06168 (2015). URL: http://arxiv.org/abs/1510.06168.

[53] Peilu Wang et al. "Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network". In: *CoRR* abs/1510.06168 (2015).

[54] Shuly Wintner. "Hebrew computational linguistics: Past and future". In: *Artificial intelligence review* 21.2 (2004), pp. 113–138.

[55] Eric Yuan. *Recursive neural network architecture*. URL: http://eric-yuan.me/wp-content/uploads/2015/06/1.jpg.