Extracting Bible Quotes from Historical Commentary

Asaf Yeshurun
Extracting Bible Quotes from Historical Commentary

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Asaf Yeshurun

Submitted to the Senate
of the Technion — Israel Institute of Technology
Iyar 5781 Haifa April 2021
This research was carried out under the supervision of Prof. Benny Kimelfeld, in the Faculty of Computer Science.

**Acknowledgements**

I would like to thank my advisor, Prof. Benny Kimelfeld, for his time, patience and professional help. I would also like to thank Dr. Oren Mishali for his generous help and advise. In addition, I would like to thank my co-workers in Prof. Benny Kimelfeld’s research lab for their feedback.

The generous financial help of the Technion is gratefully acknowledged.
Contents

List of Figures

Abstract 1

1 Introduction 3
   1-1 Related Work 4

2 Problem Statement 5

3 Techniques 9
   3-1 A Rule-Based Model 9
   3-2 A Classical Machine Learning Model 10
      3-2-1 IOB Tagging 12
      3-2-2 Selected Features 12
   3-3 A Neural Network Model 13

4 An End-to-end Solution 15
   4-1 Candidate Generation 16
   4-2 Context Extraction 17
   4-3 Candidate Scoring 17
      4-3-1 Word Embedding 17
      4-3-2 Neural Network 19

5 Experiments and Results 21
   5-1 Labeled Data 21
   5-2 Candidate Generation 22
   5-3 Different Embedding Methods 23
   5-4 Importance of Cosine Similarity 24
   5-5 End-to-End vs. Rule-Based - Across Different Corpora 24
   5-6 Rule-Based vs. Classical Machine Learning vs. Neural Network 27
      5-6-1 Neural Network Model - More in-Depth Results 27
   5-7 Case Studies 29

6 Conclusions and Future Work 31
# List of Figures

2-1 Problem demonstration. .................................................. 5

3-1 A simple example of IOB tagging ........................................ 12
3-2 Neural network structure .................................................. 14

4-1 Joint model overview ...................................................... 16
4-2 Context extracted demonstration: on the left side are the candidate quote, its sentence and its paragraph, and on the left side the candidate Biblical verse is presented, along with its chapter and its historical interpretations from various sources. ........................................ 18
4-3 Co-embedding demonstration .............................................. 19
4-4 Candidate scoring network structure .................................... 20

5-1 Tagging examples existing in Midrash Raba. Correct labels are marked in green, while a faulty label is marked in red. ......................... 22
5-2 Statistics regarding Midrash Raba and the distribution of quotations within it. ................................................................. 22
5-3 An illustration of the candidate scoring neural network component with its cosine similarity layer replaced by a standard fully connected layer with ReLU activation. ................................................. 25
5-4 Comparison between the rule-based model, the classical feature-based machine learning model and the neural network model. ............... 28
5-5 Results of the bi-LSTM neural network aimed at solving the IOB tagging problem, differing by embedding strategy. ......................... 28
5-6 An example of a paragraph perfectly tagged by the model, which even managed to correctly tag two quoted unlabeled in the preexisting labeling. 29
5-7 A second example of a paragraph tagged by the end-to-end model, including what is perhaps a quote missed by the model, and perhaps an incorrect tagging. .................................................. 29
Abstract

The Hebrew Bible (Tanach) has been extensively quoted by historical religious text and commentaries throughout history. Nowadays, many of these text resources are publicly available online. Yet, the Bible quotations within them are often partially identified if at all. Knowing the exact quotations may be highly beneficial to scholars interested in studying or investigating the Bible. We have developed and empirically analyzed several solutions to this task, utilizing both rule-based heuristics and machine-learning. End-to-end, our main model is comprised of three main stages: (a) rule-based candidate generation, (b) context extraction using available historical commentary, and (c) an artificial neural network for candidate scoring. To evaluate our models, we have constructed labeled data based on the Hebrew Bible commentary known as Midrash Raba, which contains more than half a million words and over 30,000 quotations. Our solution scores over 80% F-score, and considerably outperforms several state-of-the-art approaches for tasks of a similar nature. In addition, it scores well when tested on unfamiliar corpora in case they involve writing style and vocabulary prevalent in Midrash Raba. As a contribution of independent interest, our solution includes of a novel word-embedding method that seeks to utilize the nature of our text and its context.
Chapter 1

Introduction

Quotation detection is a well studied task with several applications, among them plagiarism detection [ES06, PSBCnR10, XFM+04], and our own task of extracting Bible quotes from historical texts [OM18, HKSS10]. Our challenge finds its roots in a central NLP discipline: named-entity recognition and classification (NER/NEC), which has been scrutinized heavily during the last decades [NS07, LBS+16, SM03, RR09, RCME11].

In this work, we develop and analyze several models that seek to identify Biblical quotations in Hebrew texts, specifically within Jewish commentaries known as "Midrash Raba", and link them to their corresponding verses. Among the models we present is an end-to-end learning based model that consists of several stages, which we consider a main contribution.

The task at hand has been tackled so far, to our knowledge, only by rule-based approaches [OM18], as well as by a relatively straightforward, classical machine learning approach [HKSS10].

Our main model envelops three stages: First - we parse the text in question and generate potential candidates, that is, n-grams that have some likelihood of being an intended quote, paired with their alleged quoted verses. Second - context extraction. For each such candidate-verse pair, we extract a large volume of context, including the candidate’s sentence and paragraph within the text in question, the verse’s chapter within the Bible, as well as commentaries discussing the verse taken from various historical sources. Third - candidate scoring. This stage is done by a neural network, and it builds upon a previous work from a similar domain [FLDK16]. It captures a known idea of utilizing context, both of the candidate quote and its alleged match in the Bible, in order to aid the network identify whether each candidate is indeed an intended quote.

Our models are trained on labeled data that we have constructed for the purpose of this research and is available online, including a text of more than 600,000 words long and over 20,000 Bible quotations. We evaluate our models using several known measures [POK+13], and under various conditions.
1-1 Related Work

The task of Biblical quotation detection in Hebrew, as we define in Chapter 2, has scarcely been tackled in research. Despite that, our challenge is closely related to several other, better studied problems within the realm of NLP, mostly NER, NEL and sequence tagging.

NER, also known as entity identification, entity chunking and entity extraction is a subtask of information extraction that seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organizations, locations, medical codes, time expressions, quantities, etc. In our case, we could define our problem as identifying the named entities of Bible quotations within a broader text, that is, extracting the parts of a given text that correspond to specific Biblical verses.

A second subtask of information extraction that is highly relevant to our work is NEL, or entity linking [RCME11, MRN14, ZSTW10]. NEL goes hand-in-hand with entity recognition, and seeks to assign a unique identity to an entity, possibly from a pre-defined knowledge base. In our work we seek not only to identify Bible quotes, but also to link between them and their corresponding Bible verses.

A third NLP task tangent to our work is sequence tagging [TSY03, YVDCBC13, PABP17]. Sequence tagging is, simply put, assigning a label (a tag) for each element in a sequence. This task is often addressed using neural networks [HXY15] with a variety of structures.

The task called quotation detection, despite being defined differently than how we define it throughout this work, has been studied in the past mainly in English [POK+13, BP07, SKP16, AAM14], but also in Hebrew [OM18, HKSS10], French [BG08] and Portuguese [FMM11].

Our task of Bible quotation detection and linking, although related to previous works, differs from them for several reasons. First, we strive to identify Bible quotation in historical religious texts completely lacking quotation marks or any other clear indication of quotes being present. This is opposed, at least in part, to the texts studied in other works [BP07, FMM11, SDS10, BG08, AAM14].

Second, our texts of interest are written in ancient Hebrew mixed with Aramaic, both lacking heavily in morphological analysis tools, thus preventing us from developing efficient models that are heavily reliant on them. This differentiates our work from many others [POK+13, BP07, FMM11, SDS10, BG08, AAM14] that focus on contemporary texts written in languages having an abundance of such tools.

Third, since we are specifically studying Bible quotations, we not only have the text in question itself to scrutinize, but also the Bible, allowing for richer models using forms of string matching. This is because the Bible has been extensively studied throughout history, thus enabling us to utilize many interpretations and commentaries from various historical periods.
Chapter 2

Problem Statement

Despite the fact that our problem is closely related both to entity detection, entity linking and quotation recognition, it is not synonymous with any of them, as it envelops unique features and characteristics. Hence, We define our problem as follows: Given two corpora - $C_{Bible}$ and $C_{Source}$, the former corresponding to the combined Biblical text and and latter corresponding to a classical Jewish commentary or historical text, find all triplets $(q, i, v_{uri})$ such that $q$ is a string in $C_{Source}$, $i$ is the index in $C_{Source}$ where $q$ starts, and $v_{uri}$ is a uri of a Biblical verse, indicating the quoted source. We refer to the construction of such triplets as annotation.

Figure 2-1: Problem demonstration.

It is important to note the difference between our task and what is known in the literature as quotation recognition [POK+13]. While our problem seeks to connect strings in a source corpus to strings in a second corpus, the problem known as quotation recognition does not attempt to link two corpora, but rather simply detect substrings in a single standalone corpus which could be classified as quotations.

Our task is unique, generally speaking, and with it come challenges often nonexistent
neither in the realm of quotation recognition as it is usually defined in existing literature, nor in the realm of entity linking. In fact, one could say it entails the difficulties of both combined. One of the main difficulties addressing quotation recognition is the need to determine the quotes’ exact spans, despite the fact that quotations marks, or any other known signs of a given text being a quote, are often non present. This is also the case with our challenge. With regards to entity linking, one of its main challenges is deciding how to handle ambiguous entities - entity mentions that match more than one candidate in the knowledge base. Solving our task requires addressing both challenges.

We provide an end-to-end solution to our problem in chapter 4 that we consider our main contribution. Independently, as part of the research process we have also had reasonably successful attempts solving a weaker variation of the problem. A factor largely responsible for the problem’s complexity is the disambiguation step of linking quotes detected to the Biblical verses they reference. The first three techniques we discuss in 3 aim to solve the problem with this step omitted for the sake of simplicity. In other words, they aim to find all pairs \((q, i)\) in \(C_{Source}\) rather than all triplets \((q, i, v_{uri})\).

**Evaluation Metrics**

One of the main challenges we have encountered throughout our work has been choosing proper evaluation metrics for our models. Since we are dealing with, in essence, a classification challenge, a method that revolves around precision and recall seems natural. However, it is not clear how we should take into account partial or incomplete matches using common measures. If, for example, a 3-gram in a historical text aims to quote a Biblical verse, yet our model has only matched the 2-gram prefix correctly, it is not trivial how such matches should be taken into account. This problem has a greater impact in longer quotes, in which it is illogical to rule out a match completely and overlook it when calculating precision and recall only because of a single incorrectly matched word, a word that may constitute merely a small fraction of the quote. Therefore we have adopted several evaluation methods, aimed to measure the performance of our models across different aspects.

The first set of measures we have adopted is what we refer to as ”Strict measures”. These are, simply put, the known measures of precision, recall and f-score, when the successes taken into account are only the n-grams matched perfectly, word-by-word, with a Biblical verse. These measures are defined as follows: Let \(pred\) be the set of all triplets \((q, i, v_{uri})\) annotated as quotations by our classifier, and let \(true\) be the set of all such triplets that represent the ground truth. Then we define:

- Strict Precision (SP) = \(\frac{|\{p \in pred \text{ such that } p \in true\}|}{|pred|}\)

- Strict Precision (SP) = \(\frac{|\{p \in pred \text{ such that } p \in true\}|}{|pred|}\)
• Strict F-score (SF) = \( \frac{2SPSR}{SP+SR} \)

In addition, we have adopted the more flexible measures defined by [HT05] and used also by [POK13], which we name "Partial precision", "Partial recall" and "Partial F-score". For the same \( \text{pred} \) and \( \text{true} \) defined earlier, let \( \text{overlap}(t,p) \) of a string \( t \) representing a ground truth quote and another string \( p \) representing a predicted quote, be the percentage of \( t \) captured by \( p \). We define:

• Partial Precision (PP) = \( \frac{\sum_{t \in \text{true}} \sum_{p \in \text{pred}} \text{overlap}(t,p)}{|\text{pred}|} \)

• Partial Recall (PR) = \( \frac{\sum_{t \in \text{true}} \sum_{p \in \text{pred}} \text{overlap}(t,p)}{|\text{true}|} \)

• Partial F-score (PF) = \( \frac{2PP \cdot PR}{PP+PR} \)

Using both strict and partial measures provides a solid framework using which we can evaluate our models more informatively.
Chapter 3

Techniques

The first solution we discuss for the weakened problem variation is a rule-based solution. It aims to identify spans of quotes using brute force string comparison, and performs candidate filtering using heuristics. Second, we discuss a classical machine learning solution which utilizes features extracted from the labeled data constructed in order to classify each word of \( C_{\text{Source}} \) using a technique called "IOB tagging", explained in 3-2-1. Overall, this model does not perform well unfortunately, mainly due to a lack of morphological analysis tools in Biblical Hebrew. The last solution we present to the weaker variant consists of a neural network that utilizes the popular bi-LSTM layers. This model is discussed in Section 3-3.

3-1 A Rule-Based Model

Approaching our task, it is common sense to attempt a straightforward solution first, both for the sake of completeness and also in order for it to serve as a baseline against which we compare more sophisticated models. In our case, a natural first approach would be a string-matching rule-based algorithm. By that we mean an algorithm that goes over the text in question and aims to find n-grams which also appear, either word-by-word or with some variation, in a Biblical verse. The skeleton of such algorithm is relatively straightforward. Yet, there are a few concepts to keep in mind, and some pitfalls to avoid. Hence, our algorithm’s steps are detailed in 1.3. Intuitively, the algorithm’s steps are as follows:

- Index the Bible using Lucene to obtain a Bible index convenient for searching.
- Format \( C_{\text{Source}} \) removing commas, full stops, and all other punctuations marks. In addition, replace the Jewish God’s name with its name used across The Bible.
- Find all bigrams in \( C_{\text{target}} \) that appear in a Biblical verse \( v \), allowing incremental Levenshtein distance, to obtain quadruplets of the form \((c, i, j, q)\) of candidate quote, index, and quoted verse.
• Go over the set of quadruplets, and merge pairs \((c, i, j, q), (c', i', j', q')\) in case \(i \geq i', j \leq j',\) and \(q = q'.\) Keep doing this process inductively until you remain with maximal length candidates. Intuitively, this step eliminates matches contained in other, broader matches for the same verse.

• Go over the set of quadruplets, and merge pairs \((c, i, j, q), (c', i', j', q')\) if \(i' \geq j + 1\) and \(q = q'.\) Keep doing this process inductively until you remain with maximal length candidates. Intuitively, this step merges overlapping matches for the same verse.

• Perform a fine-grained filtering of candidates that differ from the alleged quoted verse by a number of characters within the allowed range, however these character are not "Ehevy" characters (Hebrew vowels).

These are the main non-trivial steps taken by the rule-based algorithm. As will be demonstrated in chapter 5, this straightforward algorithm performs reasonably well, and is comparable to our machine learning models. It outperforms the classic machine learning model we present next, but proves inferior to a neural network based model.

Generally speaking, the shorter the quote - the worse this models performs, and the less likely it is to pinpoint the exact quoted verse. This is expected, since a short quote - specifically a quote comprised of two or three words, is likely to appear many times throughout The Bible, hence there is more ambiguity for the algorithm to try and resolve. This explosive ambiguity for 2-3 words long quotes, some of them appearing dozens of times in The Bible, hinders performance. This weakness inherent to rule based approaches is partly what encouraged us to seek other directions.

### 3-2 A Classical Machine Learning Model

Now that we have considered a rule based algorithm for its strengths and weaknesses, a logical step forward would be an attempt to address its weaknesses, hopefully while avoiding losing its benefits, using a more modern approach. The rule based algorithm’s main weakness is, as stated in the previous subsection, its failure to solve ambiguity in case an n-gram in \(C_{target}\) appears multiple times in The Bible. This occurs often in short quotes, 2 to 3 words long.

It is reasonable to consider machine learning in order to develop a better model and resolve, at least partly, this main weakness, since machine learning models excel at understanding the broader context. Therefore the first machine learning model we consider is a fairly standard classifier that operates using features generated and tailored by us. Yet, before we move on to detail the features generated and the precise classifier used, it is first required to detail what exactly we are classifying.
Algorithm 1.3 Rule-Based Solution

1: allowed-edit-distance-short-quotes ← 0
2: allowed-edit-distance-mid-quotes ← 1
3: allowed-edit-distance-long-quotes ← 2
4: indexed-Bible ← index($C_{Bible}$)
5: formatted-corpus ← format($C_{Source}$)
6: $(C, I, J, Q) ← \{(c, i, j, q) | c \text{ is a substring of formatted-corpus that matches an n-gram in indexed-Bible, } i \text{ is the index in formatted-corpus in which } c \text{ starts, } j \text{ is the index in formatted-corpus in which } c \text{ ends, and } q \text{ is a verse in indexed-Bible containing } c \text{ while allowing an edit distance according to the candidate’s length }\}$
7: for $(c, i, j, q)$ in $(C, I, J, Q)$ do
8:     for $(c', i', j', q')$ in $(C, I, J, Q)$ do
9:         if $(i, j)$ contained in $(i', j')$ then
10:             remove $(c, i, j, q)$ from $(C, I, J, Q)$
11:         end if
12:     end for
13: end for
14: for $(c, i, j, q)$ in $(C, I, J, Q)$ do
15:     for $(c', i', j', q')$ in $(C, I, J, Q)$ do
16:         if $(i, j)$ and $(i', j')$ overlap then
17:             if $q = q'$ then
18:                 remove $(c, i, j, q)$ from $(C, I, J, Q)$
19:                 remove $(c', i', j', q')$ from $(C, I, J, Q)$
20:                 add $(c, \min i, i', \max j, j', q)$ to $(C, I, J, Q)$
21:             end if
22:         end if
23:     end for
24: end for
25: for $(c, i, j, q)$ in $(C, I, J, Q)$ do
26:     if edit distance between $c$ and $q$ includes a change of a non "EHEVY" character
27:         remove $(c, i, j, q)$ from $(C, I, J, Q)$
28:     end if
29: end for
3-2-1 IOB Tagging

A central method that we have utilized both for our classical machine learning model and for our neural network model, is a prevalent method called IOB tagging [POK+13]. IOB tagging aims to classify each word in a corpus as one of three options: 'B', beginning of some sequence we wish to identify (in our case - a Biblical quote), 'I', inside a sequence, or 'O', outside a sequence. For example, in figure 3-1, a sample text from Midrash Raba is tagged as demonstrated.

![Figure 3-1: A simple example of IOB tagging](image)

We have chosen this method both because of its success addressing problems from the same domain [POK+13], and because it "enlarges" our dataset in the sense that, obviously, our dataset contains more words than quotes, hence classifying each word results in a "larger" dataset than classifying each quote. Exact numbers regarding the labeled data’s volume will be detailed in Section 5-1.

3-2-2 Selected Features

Using our labeled data, constructed as detailed in section 5-1, we have trained a CRF (conditional random field) classifier, a probabilistic classifier that takes into account dependencies between elements in proximity. It is fairly frequently used to tackle linguistic tasks [CXHW17, IKT05]. Basically, CRF models seek to assign probabilities to different outcomes using designated features and adjacent labels. The probability of an element $x$ belonging to each class $y$ is calculated using the following formula:

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

Alas, when seeking to generate features for a machine learning model in an aim to solve an NLP task, some natural, common features that should be generated include lexical analysis components of the word or text in question, such as POS identification, yet such morphological tools are completely lacking in Biblical or Talmudic Hebrew.

Despite this drawback, we have worked with what we have at our disposal, and have constructed for each word the following features:

- The word’s relative location within its sentence.
- The word’s relative location within its paragraph.
- The number of occurrences of the bigram containing the given word and the one after it in The Bible, allowing a Levenshtein distance of 0 edits.
- The number of occurrences of the bigram containing the given word and the one after it in The Bible allowing a Levenshtein distance of 1 edit.
• The number of occurrences of the bigram containing the given word and the one before it in The Bible, allowing a Levinshtein distance of 0 edits.

• The number of occurrences of the bigram containing the given word and the one before it in The Bible allowing a Levinshtein distance of 1 edit.

This approach, unfortunately, falls short of the rule-based algorithm presented in the previous section. It is very likely that the lack of morphological analysis tools proved to be too great a handicap. In addition, despite the CRF classifier’s ability to learn how tags of words in proximity to one another behave, the tagging provided by this method was still often "full of holes", meaning a word was tagged with 'B', following by several words tagged as 'I', then several words tagged with 'O's' and then 'I's once again, despite the fact that a sequence of 'I's without a 'B' preceding it is illegal.

Despite this model scoring reasonably well in some respects, its poor performance overall is disappointing. A more robust machine learning model was needed, one that could avoid the need of lexical analysis tools, yet prove effective at understanding context and classifying accordingly.

3-3 A Neural Network Model

The next model we have developed to solve the weaker variant of our task is a neural network model. Such models tend to excel at understanding context and taking broader perspective into account, while avoiding the need for distinct and separate features constructed by developers necessary for classical machine learning models.

A substructure that we have utilized and heavily relied on in our network structure is a bi-LSTM layer, popular in models used for various NLP purposes [YHPC18]. The output layer is a softmax layer, interpreted as probabilities of a given input word being labeled either a 'B', an 'I' or an 'O'. The softmax activation of a vector $z = (z_1...z_K) \in \mathbb{R}^K$ is defined as $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$. The full network structure is detailed in figure 3-2.

We have also experimented with several word embedding methods described later in this chapter, and the corresponding results are detailed in chapter 5.

Trial and error taught us that it is best, at least for the family of architectures we have studied, to feed each word along with its neighbors to the network separately, rather than feed the network passages one after the other. Therefore this model is fed 7 input vectors, corresponding to the word we wish to classify, along with 3 neighboring words from each side.
Figure 3-2: Neural network structure
Chapter 4

An End-to-end Solution

Finally, a main contribution of ours is an end-to-end model that combines both a rule-based string-matching approach and a neural network learning based approach in order to combine the strengths of both approaches - the high recall of the rule-based algorithm, and the deep context understanding of learning based models.

The main idea behind this model is the following: First, go over the target corpus and search strings that match the source corpus, while allowing many character swaps, additions and omissions. This stage, intuitively, aims to detect as many (if not all) quotations, and serve as a filter that leaves us with all possible candidates of quote-source pairs. If we were to stop after this stage, our model would be very strong in terms of recall, since most if not all quotes would be found, but very weak in terms of precision.

The second stage is what we refer to as context extraction. Since we aim to utilize neural networks’ ability to understand context, we strive to provide it with as much context as possible. We therefore extract for each candidate-verse pair \((c, v)\) several contextual elements, detailed in Section 4-2.

The third stage filters all context-enriched pairs of candidate and quote through a pre-trained neural network. The network aims to take a pair of candidate-quote that are likely to be similar linguistically, since the quote was detected by the rule-based stage, and determine whether or not this pair is indeed intended to be quote be the authors of \(C_{Source}\). Here we should emphasize, that even if some string in \(C_{Source}\) matches a string in \(C_{Bible}\) character-by-character, it is still not guaranteed that the author of \(C_{Source}\) intended it to be a reference at all. It could simply be a coincidence. Hence, the network aims to detect, for every candidate-quote pair, whether the author intended the candidate to be a quote, and if so, whether the quote matched with candidate is indeed the correct match. An overview of the model is presented in figure 4-1. The network provides a relevance score \(r\) as output for each context-enriched pair. \(r\) denotes the likelihood of the candidate-quote pair being a correct match intended by the author, given the context of the candidate within the target text and the context of the source quote within The Bible. After this stage, we are still left with the task of deciding
a binary answer for each such pair, and returning what has been decided as output.
This stage is done simply by filtering unlikely pairs by a likelihood threshold - every
pair with \( r < 0.5 \) we reject as there is a strong chance that it is indeed not intended
to be a quote. Otherwise, we accept. This threshold of 0.5 has proved, from trial and
error, to be the best under our current architecture. It is also the most natural. In
case of ambiguity, in other words, in case some alleged quotation \( q \) and some Bible
verse \( v \), both \((q, v)\) and \((q, v')\) are classified by the network as strong matches, only the
candidate with the higher probability is taken.

4-1 Candidate Generation

As explained previously, our joint model's first step is candidate generation. By that we
mean, given a text corpus \( C \), obtaining a list of pairs \((c, v)\) such that \( c \) is a substring of
\( C \), \( v \) is a Biblical verse, and one can suspect that \( c \) may indeed be an intended reference
to \( v \).

Before we describe our method of generating such pairs, it is important to note that
every possible candidate generation method involves some trade-off between recall and
precision. If, for example, we were to generate candidates by searching for all substrings
of length larger than 4 words that are perfectly matched (character by character) to
a substring of a Biblical verse, and are also in proximity to another substring that
matches a verse from the same Biblical chapter perfectly, we were very likely to achieve
a precision score close to 100% with this method alone. In this case we would, of course,
pay a significant price in recall. Alternatively, taking as candidate every substring of
length 2 or more that even remotely resembles a substring of a Biblical verse, without
any extra conditions, would result in close to 100% recall for the price of abysmal
precision.

With this principle in mind we have designed the candidate generation component
to be largely skewed in favor of recall. Our thought behind this decision, which proved
reasonable later, was that the task of filtering faulty candidates is more suitable to be
handled by a learning based model, since such models take broader context into account.
Hence, we have experimented with a few options for an acceptable edit distance (namely
Levinshtein distance) between \( c \) and \( v \), and the recall/precision trade-off for each option
is detailed in Section 5-2. Taking all into account, the option that turned out to be best is the incremental edit distance option, which allows for more edits the longer a candidate gets.

4-2 Context Extraction

We strive to provide each pair \((c, v)\) with as much broader context (both Biblical context for \(v\) and Midrashic/Talmudic context for \(c\)) as possible. This approach is consistent with our understanding of neural-network based models in general. Thus, while basing our approach on [FLDK16] and expanding in aspects we found useful, we have extracted for each such pair the following texts:

- The sentence \(c_s\) in our target corpus in which the candidate resides.
- The paragraph \(c_p\) in our target corpus in which the candidate resides.
- The chapter \(v_c\) in which the verse resides.
- A collection of various interpretations and discussions \(v_i\) of the verse \(v\) in classical Jewish texts throughout history.

The reasoning behind the latter is that we expect a verse’s collection of interpretations and discussions to include numerous repetitions of the verse’s words, either exact repetitions or ones slightly altered due to conjugation or changes in writing style across time. The context elements extracted in this stage are demonstrated in Figure 4-2.

Hence, once this stage is over, the network is fed pairs enriched with their contexts as described, that is, sextuplets of the form \((c, c_s, c_p, v, v_c, v_i)\), as illustrated in Figure 4-4.

4-3 Candidate Scoring

Before we describe this main component’s structure, a discussion on word embedding is in order, since neural network cannot be fed non-numeral values. This discussion is also relevant for 3-3, as it details the methods used and tested both for this end-to-end model and to the neural network model that aims to solve the weakened variant of the problem using IOB tagging.

4-3-1 Word Embedding

As in all neural-networks that analyze text, our learning-based component must include a form of word embedding - identifying each word in our corpora with a numeral vector. Due to the distinct nature of our problem, this need raises an immediate question - since we are dealing with corpora of distinct texts from different periods (The Bible, its interpretations and its commentaries), how should we utilize existing word-embedding
methods and tailor them to our needs, while taking into account the variety and richness of the vocabulary and grammatical structures used across distinct historical texts?

An approach that first comes to mind is simply constructing an embedding dictionary from the concatenation of all classical Jewish texts in our disposal (including The Bible) as if they were one large corpus, and in fact, this approach provides strong results, sometimes optimal out of all the approaches we have tried. Nevertheless, careful thinking leads to a few other ideas of combining our corpora into embedding dictionaries.

A unique method that we have developed and tailored specifically to the nature of our problem, is based on constructing two distinct word-embedding dictionaries of length $k$ each, one for The Bible alone, and one for all interpretations and commentaries combined. Using this method, we represent each word as a vector of length $2k$, with the first $k$ components matching its representation in the Biblical embeddings (or 0’s if the word does not appear in The Bible), and the last $k$ components matching its representation in the interpretations/commentaries embeddings (or 0’s otherwise, as in the first $k$ components). This method, which we call co-Embedding, is illustrated and Figure 4-3. Co-embedding is backed up by intuition, since our challenge lies in discovering and studying a connection between The Bible and other historical texts, as opposed to, for example, studying contemporary texts using both The Bible and its interpretations as one single blended unit. Moreover, our experiments, detailed in Chapter 5, prove this method to be, in many cases, better than its more natural counterpart described earlier. For each of the separate embedding dictionaries needed
to construct the joint co-embedding vector, we have utilized a standard bag-of-words algorithm, with a window of 5 words from each side.

Finally, we have also experimented with constructing word embeddings simply by using solely The Bible as a single corpus, and also by using solely its interpretations, as two different experiments.

### 4-3-2 Neural Network

As discussed previously, the task assigned to the network is the following: given a sixtuplet (initially a pair, with its four extra contextual components) \((c, c_s, c_p, v, v_c, v_i)\), decide whether that particular occurrence of \(c\) in \(C_{Source}\) is indeed an intended reference to \(v\).

Our network consists of the following components: first, a simple algebraic layer that averages each component of the tuple to a vector the size of a single word embedding. Second, a fully-connected layer for each of the 6 components separately, aimed to extract as much meaning from each such averaged vector as possible. Third, a cosine-similarity operator layer that calculates the cosine similarity score - a measure of similarity between two different vectors by calculating the cosine of the angle between them. For two vectors \(v, u\) in a vector space \(V\), the cosine similarity of \(u\) and \(v\) is calculated according to the following formula: \(\cosim(v, u) = \frac{u \cdot v}{||v|| ||u||}\). This measurement is calculated for each pair from \((c, c_s, c_p) \times (v, v_c, v_i)\). This step results in 9 different outputs, as demonstrated in figure 4-4. These similarity scores are fully connected to a 2-neuron wide output layer, with a softmax activation function. The two neurons, intuitively, correspond to the two options we would like to decide between: A good/likely match, and a bad/unlikely match, and the softmax activation, alongside the cross-entropy loss function, is interpreted as assigning a probability to the likelihood of the pair \((c, v)\) being an intended reference. We provide the mathematical formula of the softmax activation function in Section 3-3. As for the cross-entropy loss function, it is defined as follows: \(L_{CE} = - \sum_{i=1}^{n} t_i \log(p_i)\) such that \(n\) is the number of classes, \(\{t_i\}_{i=1}^{n}\) are the truth labels for each element classified, and \(\{p_i\}_{i=1}^{n}\) are the probabilities predicted by the softmax function.
Figure 4-4: Candidate scoring network structure
Chapter 5

Experiments and Results

Since we consider our end-to-end model a main contribution, most of our experiments focus on its performance. We present results regarding the performance of the candidate-generation stage alone as a function of how generous we are accepting noisy candidates, results regarding the effect of different embedding method on the model’s performance, comparisons to our other techniques that serve as baselines, difference in performance when evaluating our model on a different corpus than the one it was trained on, and more. But first, we describe the process of constructing our labeled data, which we also consider a central contribution, as well as provide some statistics about it.

5-1 Labeled Data

Our work’s initial goal was to design and evaluate a learning-based model, or a model that includes a critical learning-based component, that performs well in the realm of Biblical quotation detection in historical texts. Unfortunately, no standard labeled data of any Hebrew historical text is available, and no canonical results on such texts are available for us to compare ourselves against, at least as far as we have been able to discover.

Thus, we have constructed our own labeled data using the known historical texts "Midrash Raba”, which are commentaries on several books within The Bible: Genesis, Exodus, Leviticus, Numbers, Deuteronomy, Song of Songs, Book of Ruth, Lamentation, Ecclesiastes, Book of Esther. These sources are available as part of the combined Jewish historical knowledge. We have chosen this source since one can extract the exact locations of the intended quotes within them, along with the Biblical verses corresponding to these quotes, fairly consistently. This is due to annotations existing in the texts themselves. An example of such an annotation is presented in Figure 5-1.

References in Midrash Raba almost always start with a reference bordered by parenthesis as illustrated in Figure 5-1, and almost always end with either a comma or a full-stop. denoting an end of a quote. Using this knowledge, we have been able to algo-
Figure 5-1: Tagging examples existing in Midrash Raba. Correct labels are marked in green, while a faulty label is marked in red.

arithmically extract labeled data from Midrash Raba. Statistics regarding the volume of this labeled data and the distribution of quotes within it are presented in Figure 5-2, and we consider it a contribution of our research.

Figure 5-2: Statistics regarding Midrash Raba and the distribution of quotations within it.

5-2 Candidate Generation

As advertised in Chapter 3, we have constructed the candidate generation component in our end-to-end model so that it sacrifices precision for the sake of better recall. That is, we scan the target corpus linearly in order to find and tag all remotely-reasonable matches to a Biblical verse, and save these matches as pairs, which we enrich with context in the next component.
It is here that we show how well this component fairs by itself, and how the Levin-
shtein distance we allow between an alleged quote and a verse in order for it to be accepted as a match affects recall and precision, both strict and partial as we have defined in Section 2. The results of this experiment are detailed in Table 5-1.

It is evident by these results that, as expected, allowing greater edit distance during this stage gains diminishing returns in recall, while punishing precision.

Table 5-1: Candidate generation component recall precision as a function of the levenshtein distance allowed between an alleged quote and a Biblical verse.

<table>
<thead>
<tr>
<th>l.d Measure</th>
<th>Strict Recall</th>
<th>Strict Precision</th>
<th>Partial Recall</th>
<th>Partial Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 l.d</td>
<td>0.8372</td>
<td>0.47</td>
<td>0.884</td>
<td>0.5797</td>
</tr>
<tr>
<td>1 l.d</td>
<td>0.8501</td>
<td>0.4457</td>
<td>0.9143</td>
<td>0.5559</td>
</tr>
<tr>
<td>2 l.d</td>
<td>0.861</td>
<td>0.41</td>
<td>0.92</td>
<td>0.53</td>
</tr>
<tr>
<td>3 l.d</td>
<td>0.863</td>
<td>0.41</td>
<td>0.924</td>
<td>0.53</td>
</tr>
<tr>
<td>Inc l.d</td>
<td>0.86</td>
<td>0.4597</td>
<td>0.932</td>
<td>0.5616</td>
</tr>
</tbody>
</table>

5-3 Different Embedding Methods

As discussed in subsection 4-3-1, a key stage in the learning-based model, and one critical to the performance of our end-to-end model, is the word-embedding stage. We have experimented with several word-embedding methods, and the end-to-end model’s results on the Midrash Raba dataset when utilizing each embedding method separately is detailed in tables 5-2 and 5-3. All embedding methods were performed using the bag-of-words algorithm with a window of 5 words. Performance for each method is measured using the 6 measures discussed in 2. Each method is compared across different possible embedding dimension sizes (meaning the length of the embedding vector for each word). When discussing our novel method that we have named "Co-embedding", dimension refers to the combined length of the two embedding parts, as opposed to each of the two parts separately.

Table 5-2: End-to-end model results, differing by embedding method, with embedding dimension .100

<table>
<thead>
<tr>
<th>Measure</th>
<th>Bible</th>
<th>Commentaries</th>
<th>Bible and Combined</th>
<th>Co-Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Precision</td>
<td>0.584</td>
<td>0.767</td>
<td>0.821</td>
<td>0.799</td>
</tr>
<tr>
<td>Strict Recall</td>
<td>0.47</td>
<td>0.694</td>
<td>0.722</td>
<td>0.739</td>
</tr>
<tr>
<td>Strict F-score</td>
<td>0.52</td>
<td>0.728</td>
<td>0.7683</td>
<td>0.7678</td>
</tr>
<tr>
<td>Partial Precision</td>
<td>0.623</td>
<td>0.81</td>
<td>0.87</td>
<td>0.854</td>
</tr>
<tr>
<td>Partial Recall</td>
<td>0.565</td>
<td>0.73</td>
<td>0.77</td>
<td>0.78</td>
</tr>
<tr>
<td>Partial F-score</td>
<td>0.592</td>
<td>0.767</td>
<td>0.8169</td>
<td>0.815</td>
</tr>
</tbody>
</table>
Table 5-3: End-to-end model results, differing by embedding method, with embedding dimension .200

<table>
<thead>
<tr>
<th>Measure</th>
<th>Bible</th>
<th>Commentaries</th>
<th>Combined</th>
<th>Co-Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Precision</td>
<td>0.585</td>
<td>0.776</td>
<td>0.826</td>
<td>0.803</td>
</tr>
<tr>
<td>Strict Recall</td>
<td>0.47</td>
<td>0.698</td>
<td>0.724</td>
<td>0.744</td>
</tr>
<tr>
<td>Strict F-score</td>
<td>0.521</td>
<td>0.734</td>
<td>0.7717</td>
<td>0.772</td>
</tr>
<tr>
<td>Partial Precision</td>
<td>0.622</td>
<td>0.819</td>
<td>0.877</td>
<td>0.859</td>
</tr>
<tr>
<td>Partial Recall</td>
<td>0.567</td>
<td>0.734</td>
<td>0.7748</td>
<td>0.781</td>
</tr>
<tr>
<td>Partial F-score</td>
<td>0.593</td>
<td>0.774</td>
<td>0.8227</td>
<td>0.8181</td>
</tr>
</tbody>
</table>

5-4 Importance of Cosine Similarity

Since it is uncommon to include a cosine similarity operator in a neural network, it raises questions as to whether this operator provides contribution to the model’s performance. We have considered two main alternatives for comparison - either simply omitting this layer, and feeding the previous layer’s results, concatenated, to the last 2-neuron output layer, or replacing the cosine similarity layer with a fully connected layer, utilizing a standard ReLU activation, defined as: $f(x) = max(0, x)$. An illustration of this potential replacement is presented in figure 5-3. This experiment’s results are detailed in table 5-4.

Table 5-4: End-to-end model results when modifying the cosine-similarity layer within the candidate scoring component. "Unchanged" denotes leaving this layer as-is, and "FC" denotes Fully-Connected.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Omitted</th>
<th>FC (dim 1024)</th>
<th>FC (dim 2048)</th>
<th>Unchanged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Precision</td>
<td>0.663</td>
<td>0.773</td>
<td>0.777</td>
<td>0.803</td>
</tr>
<tr>
<td>Strict Recall</td>
<td>0.515</td>
<td>0.699</td>
<td>0.705</td>
<td>0.744</td>
</tr>
<tr>
<td>Strict F-score</td>
<td>0.578</td>
<td>0.734</td>
<td>0.739</td>
<td>0.772</td>
</tr>
<tr>
<td>Partial Precision</td>
<td>0.712</td>
<td>0.843</td>
<td>0.846</td>
<td>0.859</td>
</tr>
<tr>
<td>Partial Recall</td>
<td>0.565</td>
<td>0.722</td>
<td>0.729</td>
<td>0.781</td>
</tr>
<tr>
<td>Partial F-score</td>
<td>0.63</td>
<td>0.7778</td>
<td>0.783</td>
<td>0.8181</td>
</tr>
</tbody>
</table>

5-5 End-to-End vs. Rule-Based - Across Different Corpora

Despite the fact that the rule-based algorithm discussed in section 3-1 and the end-to-end model discussed in Chapter 4 do not solve the exact same problem, since the rule based algorithm does not seek to disambiguate in case several matches are found for a potential candidate. Hence, the rule-based algorithm was not evaluated in exactly the same way as the final, end-to-end model. Its evaluation only takes into account the
Figure 5-3: An illustration of the candidate scoring neural network component with its cosine similarity layer replaced by a standard fully connected layer with ReLU activation.
spans identified as delineating quotes compared to the ground truth delineations (i.e.
spans), disregarding whether a quote associated with a candidate is correct. This is
a key observation since many intended quotes in the historical texts we analyze have
several exact matches. Therefore, results obtained when “downgrading” our end-to-end
model’s measurement’s strictness, are expected to be better.

Another key aspect to address is the end-to-end model’s ability to generalize its
understanding to corpora different from the ones it was trained on. As one would
think intuitively, the more a corpus in question deviates from Midrash Raba - on which
the end-to-end model was trained, the less effective the model becomes. This effect
is irrelevant for the end-to-end model, since it does not involve any learning. Yet, a
difference in corpus does have some effect, since the algorithm was optimized on the
macro level with Midrash Raba as its labeled data. Hence, the comparison between our
end-to-end model and the rule-based model, taking into account the aspects we have
described in this subsection, is detailed in tables 5-5, 5-6, 5-7.

Table 5-5: A comparison between the rule-based heuristic model and the end-to-end
model on Midrash Raba.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Rule-based</th>
<th>End-to-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Precision</td>
<td>0.4313</td>
<td>0.827</td>
</tr>
<tr>
<td>Strict Recall</td>
<td>0.4342</td>
<td>0.759</td>
</tr>
<tr>
<td>Strict F-score</td>
<td>0.4327</td>
<td>0.791</td>
</tr>
<tr>
<td>Partial Precision</td>
<td>0.6787</td>
<td>0.865</td>
</tr>
<tr>
<td>Partial Recall</td>
<td>0.6741</td>
<td>0.77</td>
</tr>
<tr>
<td>Partial F-score</td>
<td>0.6763</td>
<td>0.814</td>
</tr>
</tbody>
</table>

Table 5-6: A comparison between the rule-based heuristic model and the end-to-end
model on Midrash Raba, when training the candidate scoring component of the
end-to-end model on 9 of the 10 parts, and testing on the remaining part. This
experiment has been run 10 times, and the numbers in this table are an average.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Rule-based</th>
<th>End-to-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Precision</td>
<td>0.4315</td>
<td>0.792</td>
</tr>
<tr>
<td>Strict Recall</td>
<td>0.4343</td>
<td>0.726</td>
</tr>
<tr>
<td>Strict F-score</td>
<td>0.4328</td>
<td>0.7575</td>
</tr>
<tr>
<td>Partial Precision</td>
<td>0.676</td>
<td>0.838</td>
</tr>
<tr>
<td>Partial Recall</td>
<td>0.674</td>
<td>0.791</td>
</tr>
<tr>
<td>Partial F-score</td>
<td>0.675</td>
<td>0.813.0</td>
</tr>
</tbody>
</table>
Table 5-7: A comparison between the rule-based heuristic model and the end-to-end model on Talmud Bavli, a Talmudic text vastly different in style than Midrash Raba on which the end-to-end model was trained.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Rule-based</th>
<th>End-to-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Precision</td>
<td>0.3546</td>
<td>0.101</td>
</tr>
<tr>
<td>Strict Recall</td>
<td>0.3703</td>
<td>0.157</td>
</tr>
<tr>
<td>Strict F-score</td>
<td>0.36228</td>
<td>0.1229</td>
</tr>
<tr>
<td>Partial Precision</td>
<td>0.447</td>
<td>0.262</td>
</tr>
<tr>
<td>Partial Recall</td>
<td>0.429</td>
<td>0.195</td>
</tr>
<tr>
<td>Partial F-score</td>
<td>0.4378</td>
<td>0.2235</td>
</tr>
</tbody>
</table>

5-6 Rule-Based vs. Classical Machine Learning vs. Neural Network

The results of the three models aimed to solve the quotation detection problem (rule-based, classical machine learning and neural network) are presented in Figure 5-4. These models are comparable, since at the bottom line they solve the same problem - the weaker variation aiming to only delineate quotes, not taking into account which verses are quoted. Unfortunately, the handicap of morphological analysis tools in Biblical or Talmudic Hebrew not existing proved too great, and the classical machine learning model does not provide good results overall. The neural network works better, and does manage to internalize context, even outperforming the rule-based natural counterpart.

5-6-1 Neural Network Model - More in-Depth Results

As stated, The neural network model achieved the best results out of all 3 models aimed to solve the quotation detection problem. As suspected, it seems that this model utilizes well learning based models’ ability to understand context, and scores well when tested on our Midrash Raba dataset. This model’s results, differing by embedding strategy, are presented in Figure 5-5.

In addition, more in depth results are presented in figure 5-8, detailing tagging statistics for each of ‘T,’ ‘O,’ ‘B’ separately for the best form of this neural network model, achieved when utilizing co-embedding, the word embedding variant we have discussed in Subsection 4-3-1.

Table 5-8: More Statistics of the bi-LSTM neural network on the Midrash Raba test set.

<table>
<thead>
<tr>
<th>IOB</th>
<th>Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0.87</td>
<td>0.63</td>
<td>0.73</td>
<td>5,190</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0.78</td>
<td>0.81</td>
<td>0.79</td>
<td>23,150</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>104,651</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5-4: Comparison between the rule-based model, the classical feature-based machine learning model and the neural network model.

Figure 5-5: Results of the bi-LSTM neural network aimed at solving the IOB tagging problem, differing by embedding strategy.
5-7 Case Studies

To conclude this chapter, we present two paragraphs tagged by our best, end-to-end model as examples, in figures 5-6 and 5-7. It is important to note that the preexisting tags in blue text were not available, of course, to the model, and are only presented here for the sake of clarity. A second important note - while the first paragraph was tagged perfectly, our model missed a quote in the second paragraph. However, it is unclear whether to consider this a flaw of the model or an incorrect preexisting labeling, since the two quoted words, despite appearing both in the quoted verse, do not appear adjacent to one another. The two tagged paragraphs are presented in figures 5-6 and 5-7.

Figure 5-6: An example of a paragraph perfectly tagged by the model, which even managed to correctly tag two quoted unlabeled in the preexisting labeling.

Figure 5-7: A second example of a paragraph tagged by the end-to-end model, including what is perhaps a quote missed by the model, and perhaps an incorrect tagging.
Chapter 6

Conclusions and Future Work

We have constructed and evaluated several models for the identification and classification of Biblical quotes within historical texts. As part of our end-to-end model we have utilized a learning-based component that provides significant benefits in an attempt to classify marginal matches correctly, and decide which Bible verse is the one most likely quoted. This idea proved useful, as our end-to-end model scores over 81% partial F-score and over 77% strict F-score.

We have also constructed labeled data of large volume from the Midrash Raba corpora in order to evaluate our models, consisting of more than 600,000 words with over 20,000 quotations. We present this aspect as a second contribution.

Despite the success of the end-to-end model, it in an interesting subject of future research whether this problem can be solved end-to-end by a single neural network and achieve state of the art results without the need of extra help provided by a string matching component. This question is yet unanswered.

Such a model, in order to to perform the entire process by itself, would require labeled data of very large scale, much larger than the Historical Hebrew texts at our disposal. This is because in order to classify each relevant n-gram as a verse, one would need an entire class to represent each verse, and taking into account that The Bible contains 23,204 verses (in the Hebrew canonical version), there would be a need for 23,204 classes to consider for each candidate quote. We do not have labeled data large enough to consider such models.

Therefore, future work in this subject may include experimenting with different variations of the quotation recognition and quotation classification problem, or constructing labeled data of much larger scale, such that a state of the art fully learning-based model could be feasible.

Other than that, it is interesting to explore further how different neural network structures could improve the learning-based component of the end-to-end model we currently use, and how modern breakthroughs in NLP-aimed neural networks can be utilized to further improve our results. Despite the fact that a portion of our model’s inaccuracies and mistakes is due to the labeled data itself being lacking in annotation
and faulty, we believe there is still room for improvement.
Bibliography


[HT05] Bill Hollingsworth and Simone Teufel. Human annotation of lexical chains: Coverage and agreement measures. In ELECTRA Workshop on Methodologies and Evaluation of Lexical Cohesion Techniques in Real-world Applications (Beyond Bag of Words), page 26, .2005


In the final model, we evaluated the performance of different prediction models. Specifically, we divided the predictions into two categories: (1) absolute predictions, which are considered to be perfect up to a certain point, and (2) partial predictions, which give a score for each partial match, according to the degree of overlap.

The results of the model show that the partial predictions are more accurate, with an accuracy of 86%, recall of 78%, and precision of 80%. This is higher than the scores of other known methods, which range from 70% to 80%.

To address this problem, we introduced a new method for matching words using word embeddings. This method takes a textual input, converts it into a vector representation, and then uses a classifier to predict the most likely word from a vocabulary. This approach is advantageous because it can handle words that are not in the vocabulary, and it can also handle variations in spelling and other factors.

embedding word

Throughout the thesis, we consider how different methods can be used to improve the results of these methods. We conclude that the partial predictions are more accurate and are therefore preferred for real-world applications.
Levinstein distance. When read-to-read a text for transitions between words in the text and the end of the sentence. The difference in the number of characters is called "Levinstein distance" or Distance.

Stage two—expanding all candidates, together with the sentences and the context of the Levinstein sentence at the top of the text, and the context of the Levinstein sentence at the bottom of the text. To make a decision if the text at the top of the text matches the context, and if so, this is the intended quote and the author intended, we can use the context of the other one, too—whether in the text or in the sentence context. For a candidate, we take into account additional factors, which we call the Levinstein context:

- The last sentence of the text
- The last paragraph of the text
- The last page of the text
- The last sentence of the text is a quote and the author intended.

So, at the end of stage two, we get a score for each candidate, and then we use another model to make a decision if it is a quote and if so, it is an intended quote and the author intended.

Our model uses a deep neural network with three layers: a word vector layer, a sentence vector layer, and a text vector layer. The last layer is a fully connected layer, which uses a cosine similarity to calculate the similarity between sentences and contexts, and then selects the most similar sentence as the quote.

Results

In order to test the performance of our model, we used a corpus of several thousand quotes from the famous Torah and the five books of the Bible. The quotes were tagged in a natural language processing (NLP) model, and then we used our model to identify the quotes and their context.

The results show that our model is able to accurately identify quotes and their context with high precision and recall. In addition, our model is able to handle quotes from different sources, such as the Bible, the Talmud, and other religious texts.

Conclusion

This research demonstrates the potential of using deep learning models to accurately identify quotes and their context in text. Our model was able to achieve high performance in identifying quotes and their context, and it has the potential to be used in various applications, such as text classification, question answering, and natural language generation.
התקציר

התנ"ך הוא ספר מנופח ביותר בשחיסותיו, ובתחום זה ידועו כה אינטנסיביות וסתיווניות, חל הבדלים תמורות, אשר הת.nanoTimeונותnaissanceו בשתי חלונות, ולכד מתאימים מודירים, ספירת פעולות. כיוון, ב çalışıyor החסות וסיבות התנ"ך בתנ"ך ולא מתאימים פסוקים ממגוון הכנסים בכמה זוויות ביצירת, וכנ filmer בתוככות התנ"ך tổngום

בוסתן למקורות בתנ"ך, אך גם למקורות בתנ"ך אחרים, ניתן להגדיר את התנ"ך כספר משותף בין המורות, ובו ניתן למצוא גם את המקורות התנ"כיים המופיעים במקורות אחרים. אם כי, ניתן למצוא גם את המקורות התנ"כיים המופיעים במקורות אחרים, ניתן למצוא גם את המקורות התנ"כיים המופיעים במקורות אחרים.

למרות זאת, כל המורות התנ"כיים יתנ"כיים בתנ"ך מופיעים בתנ"ך, והם מופיעים בתנ"ך, אם כי, ניתן למצוא גם את המקורות התנ"כיים המופיעים במקורות אחרים. אם כי, ניתן למצוא גם את המקורות התנ"כיים המופיעים במקורות אחרים.

 yargı.Debugger (2015), במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"ך, במאמר על התנ"כ
המחקר שעצע ב cohorte כדי פרופסור בני מיכלפלד, בפקולטה למדעי מחשב.

תודה

אני רוצה להודות ללהיות של פרופסור בני מיכלפלד, על חומן ובסבלנות רבה שאותה הקדיש, כי אם מזג על העמותת התקופה והמודעות. אני רוצה להודות לדוקטור אורן משאליצי על עוזתו המ.returnValueית במחקרים על העבירות המופתמות, ולזכות את להודות להזדמנות הנספים בציבור של פרופסור בני מיכלפלד על עוצמתו והפרידוקס החשובים.

אני מודה לסכין על המבנה הסכמתי הגרבייב הרצתי במחקרים.
יזרי ציטוטים בנ"לרים במקורות היסטוריים

היבר על מחקר

לשם ملي מי חלקי של הדרישות לכניסת התותח
מניסים עד הדגש במпромышлен

אסף ישורון

הוגה לנטה הטכנולוגיה – מרכז טכנולוגיה לישראל
איחוד החשמל א"א חיפה אפריל 2021
זיהוי ציטוטים תנ"ך במקורות היסטוריים

אסף ישורון