Title
Learning from Sequences in Education, History, and Natural Language

Author
Alkaoud, Mohamed

Publication Date
2021

Peer reviewed|Thesis/dissertation

Learning from Sequences in Education, History, and Natural Language

By

Mohamed A. Alkaoud

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Mairaj U. Syed, Chair

Kenji Sagae

Emily R. Merchant

Committee in Charge

2021

Contents

List of Figures

List of Tables

Abstract

Learning from Sequences in Education, History, and Natural Language

One of the most powerful ideas in natural language processing is the distributional hypothesis which indicates that words with similar distributions tend to have similar meanings. This led to huge movement in learning from word sequences and as a result sequential-based learning is considered one of main tools in natural language processing today. In fact, many of the recent breakthroughs in natural language processing (Word2Vec, BERT, ...) learn by exploiting sequential properties of natural language.

In this dissertation, we further explore learning from sequences more and push the boundaries on what can be learned solely from sequences. We investigate different sequences in diverse settings, ranging from educational and historical sequences to sequences of morphologically rich languages. These investigations provide us with insights and answers to questions such as: 1) can we learn good representations of non-linguistic items from their sequences?, 2) is it possible to create state of the art natural language processing models by simply rethinking sequences?, and 3) is learning an end-to-end named-entity disambiguation/entity linking system entirely from sequences feasible? Answers to these questions enlighten the machine learning, natural language processing, and computational linguistics communities on the potential, yet to be harnessed, in sequences.

Chapter 1

Introduction

The distributional hypothesis [41, 35], a basis of modern word embedding techniques, suggests that words that appear in similar contexts tend to have similar meanings: for example, given a corpus of documents, the words, 'dog' and 'cat' will appear in similar contexts and, hence, have similar meanings. Word embedding methods, operationalize the distributional hypothesis in order to produce numerical vectors as representations of words as shown below:

$$\begin{aligned} dog &= [1.36, -0.23, 3.12, \cdots] \\ cat &= [1.59, 0.04, -1.02, \cdots] \\ cab &= [-2.46, 4.25, -2.12, \cdots] \end{aligned} \tag{1.1}$$

In a seminal paper, Mikolov et al. [58] popularized word embeddings when they introduced Word2Vec and showed that it produces meaningful rich word representations, which was always a challenge for the natural language processing community. Figure 1.1 shows an overview of how these embeddings/representations are trained. Each row of squares in Figure 1.1 represents a vector. One of the things that made word embeddings popular is that they are unsupervised; they just require a large sequence of words with no additional information or extra knowledge. Moreover, the representations produced by such techniques capture interesting semantics; one popular example is how the produced vectors capture relationships between words. For example, if we subtract the vector for the word 'man' from vector of the word 'king', and then add the vector of the word 'woman' we get very close to

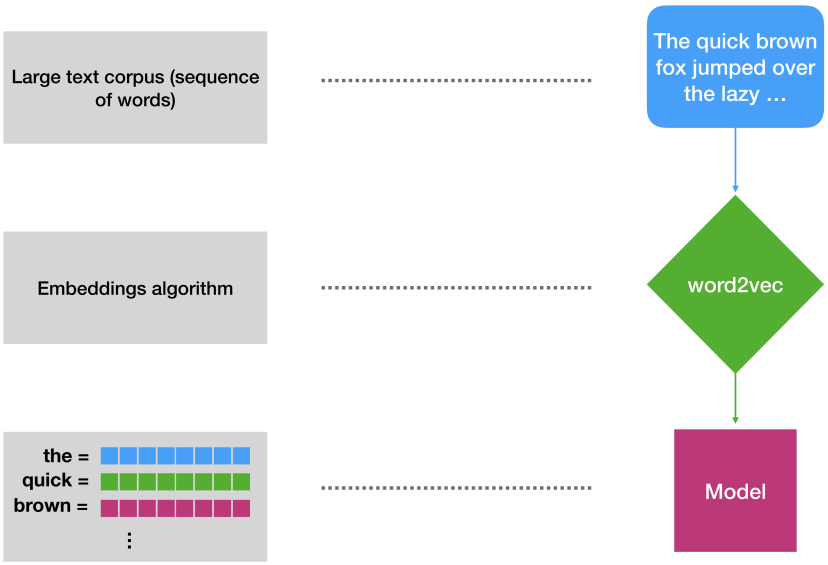

Figure 1.1: The process of training word embeddings.

the vector of the word 'queen'. Other interesting examples, relating to geography, grammar, and universities, are shown below:

$$\begin{aligned} \text{London} &\approx \text{Paris} - \text{France} + \text{England} \\ \text{swimming} &\approx \text{going} - \text{go} + \text{swim} \\ \text{UMass Amherst} &\approx \text{UC Davis} - \text{California} + \text{Massachusetts} \end{aligned} \tag{1.2}$$

We can also measure similarities between words since every word is a vector. For example, the vector of the word 'cat' will be closer to the vector of the word 'dog' than it is to the vector of the word 'cab' even though that the word 'cat' is phonetically more similar to 'cab' than 'dog'.

Sequence driven word embeddings are today considered one of the main pillars of modern natural language processing which led many scientists to research them in more detail, ranging from algorithmic-focused improvements such as better ways to train them [59, 65, 17, 47], exploiting new ways to represent them [78, 66, 29, 84], and utilizing them in information retrieval [91] to more social-centric issues such as understanding their biases [19, 23] (e.g. associating nurse with female pronouns and doctor with male ones) and employing them

as a tool to study culture [49]. Word embeddings have also been applied to many other fields such as software engineering, [30] in which researchers have exploited the locality of software [6, 42] to create vector representations of software constructs [82, 7, 40]. They have been also used in medical and clinical texts [48, 37, 50].

Unfortunately, while there has been a huge interest in word embeddings and their applications, there has not been a focus on exploring learning from sequential properties in different settings in natural language processing. Word embeddings have shown us that learning from sequences can be a very powerful tool. It seems logical that we should explore them more in natural language processing and computational linguistics.

In this dissertation, we further explore sequential based learning and push the boundaries on what can be learned from sequences. This dissertation discusses how we can learn from sequences in four different settings. First, we show that we can use sequences to learn from non-linguistic entities. We propose a sequence-based approach to generate representation of academic degrees. The second setting concerns sequences in history: we learn an end-to-end entity linking system that automatically identifies narrators in historical classical Arabic texts. The third and fourth settings relate to sequences in language. In the third setting, we learn better word embeddings by utilizing the richness of words' structures in highly inflected languages to decompose words into more granular sequences. Lastly, we explore the sequences resulting from BERT's [29] subword tokenizer, show that they are not optimal, and propose a simple way to tackle this issue. While the experiments in the third and fourth setting focus on Arabic, the techniques we propose can generalize to other highly inflected languages.

Two ideas that we strongly believe in are 1) diversity, and 2) simplicity. One manifestation of these ideas is the problems we try to tackle when doing research. All the problems tackled in this dissertation relate to diversity and NLP: Chapters 3 and 4 focus on applications of natural language techniques in fields where NLP is not commonly used: education and history. Chapters 5 and 6 focus on solutions that explore a property (morphology) that is not important in English and other common European languages since many of them are

morphologically poor [70]. The same can be said about simplicity: we try to propose simple solutions when tackling our problems. For instance, our state-of-the-art model we discuss in Chapter 6 does not require any pretraining. This is not only important due to the many benefits inherently found in simplicity, but also because it goes against the current trends of machine learning and natural language processing communities that push for more complex models neglecting the negative impact money-wise, time-wise, and environment-wise of these huge models [80].

1.1 Summary of Contributions

In this dissertation, we explore different settings where we learn from sequences. Our contributions can be summarized as:

1.1.1 Learning from Sequences in Education: Representing Academic Degrees

- We propose a way to learn representations of educational degrees from sequences of courses and show the richness of these representations by showing examples of academic degree analogies.
- We create an automatic degree curriculum contraction method based on the degree embeddings we propose and illustrate its capabilities by evaluating it on academic majors and minors.

1.1.2 Learning from Sequences in History: Identifying Narrators in Classical Arabic Texts

- We build the first automatic narrator identification system, to the best of our knowledge, for classical Arabic texts.
- We use our system to find potential mistakes in one of the largest corpora of annotated classical Arabic texts.

- We show the effectiveness of utilizing sequences in learning end-to-end entity linking systems.

1.1.3 Learning from Sequences in Natural Language: Incorporating Morphology in Traditional Word Embeddings

- We create new morphological-based word embeddings that are smaller in size, better in performance, and have superior out-of-vocabulary handling than traditional word embeddings.

- We propose a novel algorithm for generating word vectors from subword vectors.

1.1.4 Learning from Sequences in Natural Language: Rethinking Tokenization and Word Segmentations

- We investigate word segmentation in BERT and showcase issues with current word segmenters.

- We propose two NLP models: MorphBERT and CharBERT to avoid these word segmentation issues. Our models achieve state-of-the-art performance on two NLP datasets without requiring pretraining. In addition to that, they can be layered on top of any existing pretrained BERT model.

Chapter 2

Background

We will start by giving an overview of the most popular word embedding techniques. First, we will detail two traditional word embeddings: Word2Vec and fastText. After that, we will discuss contextual word embeddings and go over the most popular such technique: BERT (Bidirectional Encoder Representations from Transformers).

2.1 Traditional word embeddings

2.1.1 Word2Vec

Word2Vec introduced two main techniques to learn word representations: Skip-gram and Continuous Bag of Words (CBOW). In Skip-gram the goal is to predict the surrounding words given the word itself as shown in Figure 2.1, and in CBOW, the goal is to predict a word given its surrounding words as shown in Figure 2.2

Skip-gram

When given a word, we get representations by predicting its surrounding words. For example, if we have the following corpus: " the brown *fox* jumped over ...", the probability of 'brown' given 'fox' (P(brown|fox)) should be high. More generally, the conditioned probability on any word w should be high when it is of a word that appears in the context of w:

$$P(w_{t+i}|w_t) \tag{2.1}$$

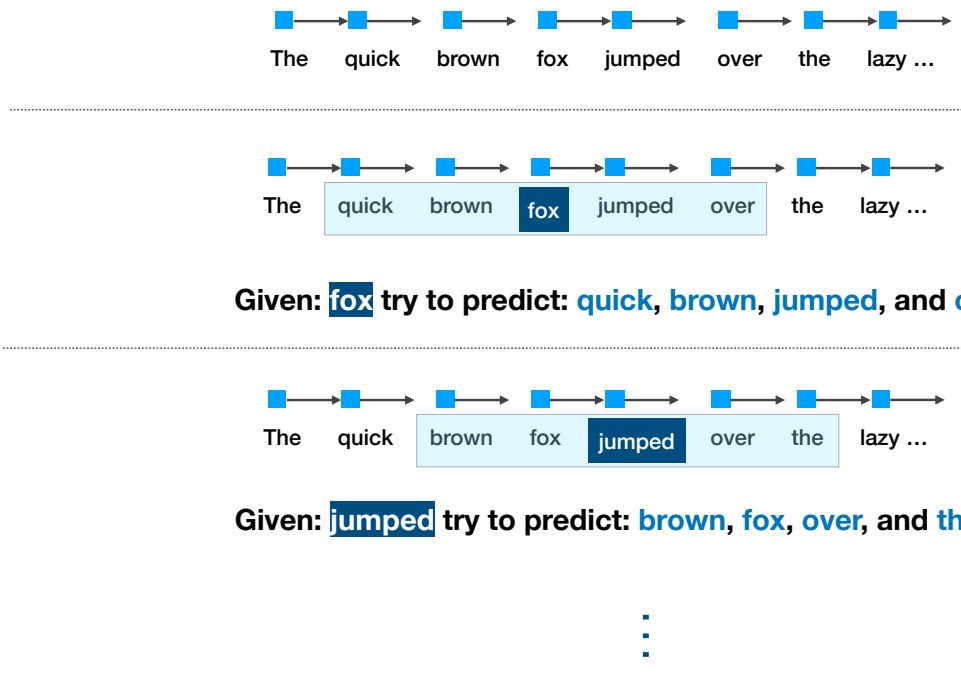

Figure 2.1: Illustration of Skip-gram.

where w_t is the word occurring at position t in our corpus. Now let's define the following function that computes the product of all such probabilities that appear in our corpus:

$$L(\theta) = \prod_{t=1}^{T} \prod_{-m \leq i \leq m} P(w_{t+i}|w_t) \tag{2.2}$$

where T is the number of words in our training corpus and m is the window size we are using and θ represents all model parameters and in this case will include two matrices (u and v) as we will see later. Our goal is to find the parameters θ (the word vectors) that maximize the function $L(\theta)$:

$$\arg\max_{\theta} L(\theta) = \arg\max_{\theta} \prod_{t=1}^{T} \prod_{-m \leq i \leq m} P(w_{t+i}|w_t) \tag{2.3}$$

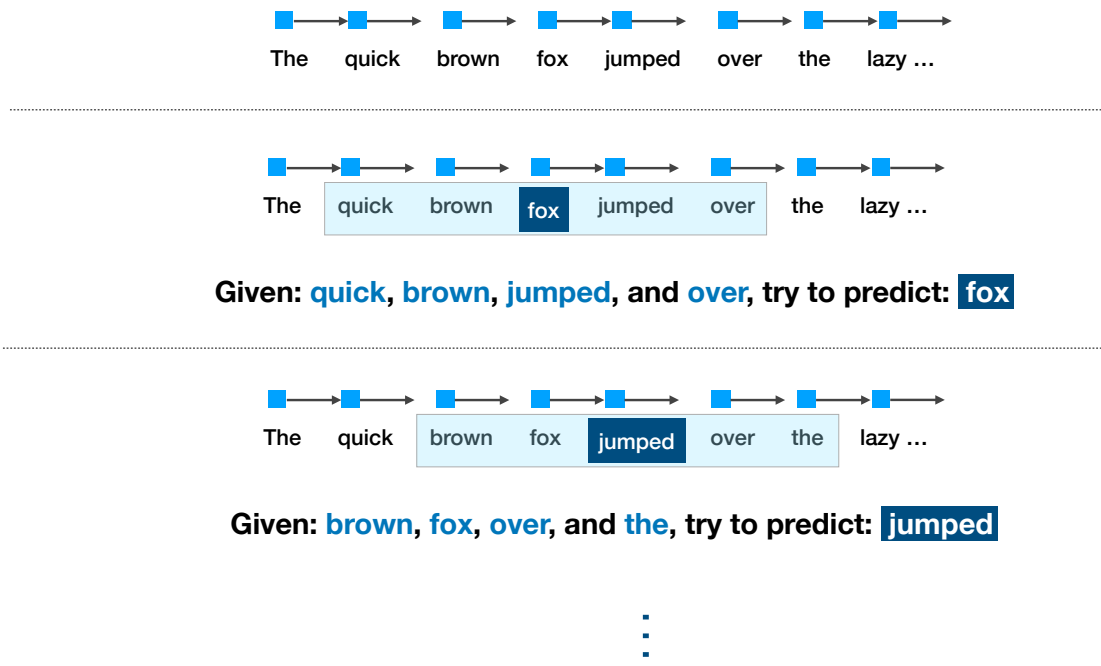

Figure 2.2: Illustration of CBOW.

Taking the logarithm will not change the maximum and will make the computation easier. We will define $J(\theta)$ to be the following:

$$J(\theta) = \log(L(\theta)) = \sum_{t=1}^{T} \sum_{-m \leq i \leq m} \log P(w_{t+i}|w_t) \tag{2.4}$$

That will give us:

$$\arg\max_{\theta} J(\theta) = \arg\max_{\theta} \sum_{t=1}^{T} \sum_{-m \leq i \leq m} \log P(w_{t+i}|w_t) \tag{2.5}$$

When optimizing in machine learning we often minimize a loss function so we will minimize the negative of our quantity instead of maximizing it:

$$\arg\min_{\theta} -J(\theta) = \arg\min_{\theta} -\sum_{t=1}^{T} \sum_{-m \leq i \leq m} \log P(w_{t+i}|w_t) \tag{2.6}$$

How are we going to compute the probabilities? We will use the following approximations:

$$P(a|b) = \frac{\exp u_a^T v_b}{\sum_{w \in V} \exp u_w^T v_b} \tag{2.7}$$

where u and v are two matrices that include the representations of our entire vocabulary. We learn two representations: u and v contain context and center representations respectively (u_w is w's representation when it occurs as a context word and v_w is its representation when it occurs as a center word). We take the exponential of the dot product of the two vector representations of words a and b and then divide by the sum of the exponential of the two vector representations of all words in our vocabulary V. Keep in mind that the dot product of the transpose of a vector and another vector is basically the summation of the product of every dimension:

$$u_a^T v_b = uv = \sum_{i=1}^{n} u_i v_i \tag{2.8}$$

Let's rewrite our optimization:

$$\arg\min_{\theta} - \sum_{t=1}^{T} \sum_{-m \leq i \leq m} \log \frac{\exp u_{w_{t+i}}^T v_{w_t}}{\sum_{w' \in V} \exp u_{w'}^T v_{w_t}} \tag{2.9}$$

So how do we optimize this quantity to find the values of u and v? We will use derivatives to analyze the rates of change and iteratively change the quantities of our parameters by adjusting them according to their rate of change:

$$\theta = \theta - \alpha \nabla_\theta J(\theta) \tag{2.10}$$

where α is the learning rate and $\nabla_\theta J(\theta)$ is the gradient. We use α in order to move in small steps and not overshoot the minimum. Now we can keep calculating the gradient and adjusting the parameters as follow:

---

while *true* **do**

$grad = \text{calculate\_gradient}(\theta, J, \text{corpus})$

$\theta = \theta - \alpha \cdot grad$

After a predetermined number of iterations or after the improvement in the loss function is negligible we can break from the loop and return our parameters θ. We then take the average of u and v to be our learned embeddings and return it.

There's a problem with the above algorithm: the $J(\theta)$ is a function of all words in our corpus which makes computing $\nabla_\theta J(\theta)$ expensive, which leads us to make infrequent parameter updates. So instead, we sample a couple of words from the corpus, calculate the gradient, and then update as shown below.

---

while *true* **do**

$sample = \text{get\_sample}(\text{corpus})$

$grad = \text{calculate\_gradient}(\theta, J, sample)$

$\theta = \theta - \alpha \cdot grad$

There's another problem that will slow down the algorithm. If we look again at equation 2.7, we notice that for each probability we have to go over all the words in our corpus which is expensive. One way to solve this is with a technique called negative sampling. In negative sampling, instead of going through all the words, we get k negative random samples and try to maximize $P(w_{t+i}|w_t)$ and minimize the probability that a random word will appear with our center word: $\sum_{k=1}^{K} P(w_k|w_t)$. One of the disadvantages of both Skip-gram and CBOW, discussed in the next paragraph, is that they cannot produce representations for words that have never been seen before in their training corpora.

CBOW

In CBOW, the goal is to predict a word given its surrounding words. Given the following corpus: "he ordered a strawberry banana *mango* protein smoothie.", the probability of 'mango' given words next to it:

$$P(\text{mango}|\text{strawberry}, \text{banana}, \text{protein}, \text{smoothie})$$

should be high as shown below:

$$\arg\max_\theta L(\theta) = \arg\max_\theta \prod_{t=1}^{T} P(w_t|w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m}) \tag{2.11}$$

The rest of the derivations are similar to the Skip-gram case.

2.1.2 FastText

FastText [47, 18, 46], a technique developed by Facebook, aims to fix the out-of-vocabulary problem with traditional word embeddings. It does so by modifying the optimization function. Let's revisit these two equations:

$$\sum_{t=1}^{T} \sum_{-m \leq i \leq m} \log P(w_{t+i}|w_t) \tag{2.12}$$

$$P(a|b) = \frac{\exp s(u_a, v_b)}{\sum_{w' \in V} \exp s(u'_w, v_b)} \tag{2.13}$$

where the $s()$ is a similarity function that we defined as the dot product of the transpose of the first vector with second: $s(u_a, v_b) = u_a^T v_b$. FastText modifies this similarity function to be:

$$s(w', b) = \sum_{g \in G_w} z_g^T v_b \tag{2.14}$$

where G_w is a set of all the n-grams of a certain size. For example setting the n to be equal three will give us the following n-grams of the word <where>[1]: <wh, whe, her, ere, re>. The model will be trained in a similar fashion to what we mentioned in the previous sections. After training, a new word r that does not appear in the training corpus can be represented as follows:

$$r = \frac{1}{|G_r|} \sum_{g \in G_r} z_g \tag{2.15}$$

2.2 Contextual word embeddings

In traditional embeddings such as Word2Vec, each word is encoded in a fixed representation of a vector. This may cause problems since many words can have multiple meanings depending on the context; *bear* in "The right of the people to keep and *bear* arms shall not be infringed." and "A wild *bear* was seen in Davis, CA." have two different meanings. Yet, traditional embeddings will only capture one fixed representation. Contextual

[1] < and > denote the beginning and ending of the word respectively.

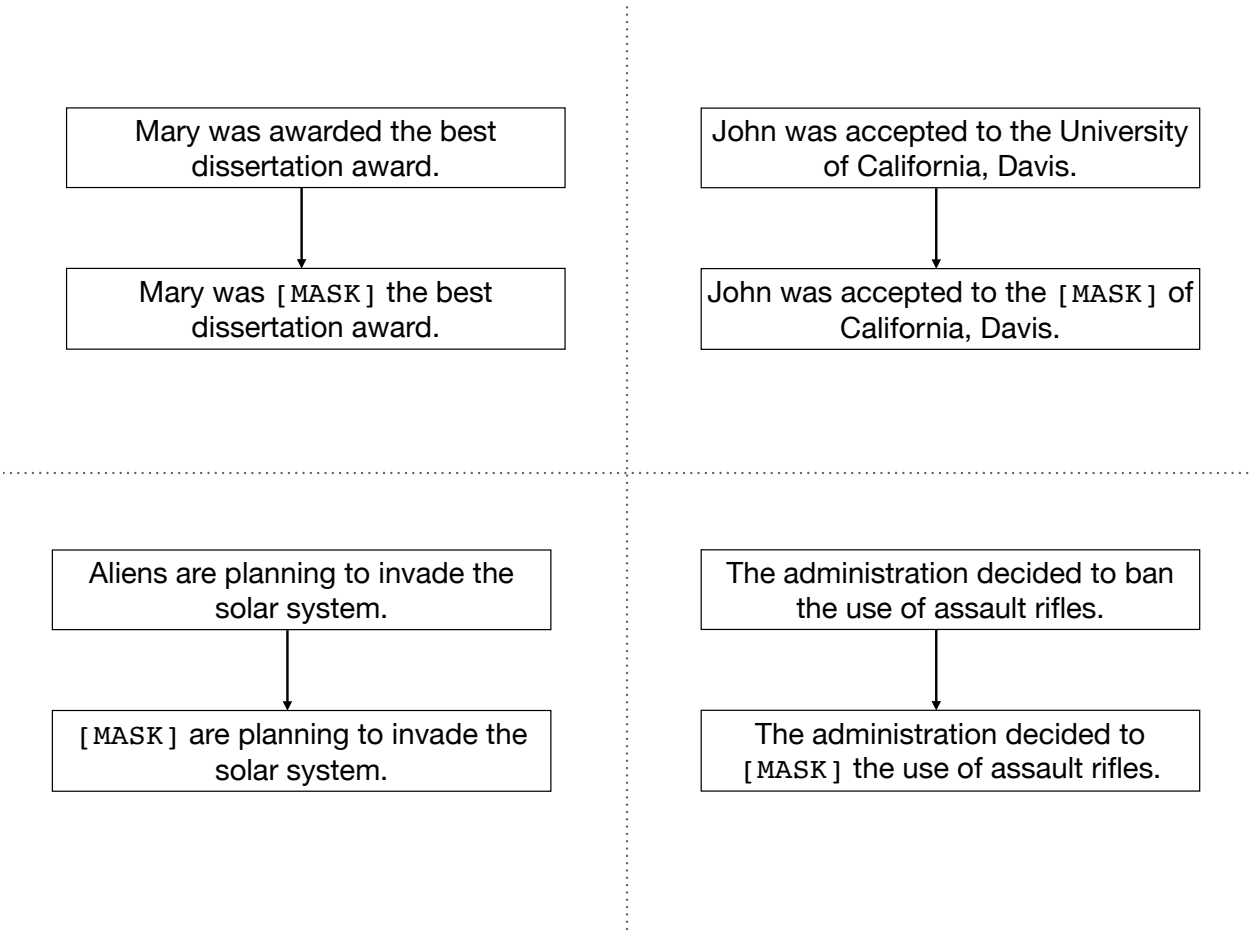

Figure 2.3: Examples of token masking in BERT.

word embeddings aim to solve this issue by modeling embeddings where the context is taken into consideration when generating the embeddings. Traditional embeddings map words to vectors: $f(word) \mapsto \mathbb{R}^n$. Contextual embeddings, on the other hand, map a word given a context to a vector: $f(word, context) \mapsto \mathbb{R}^n$. For example:

$$f(bear, \text{“... and } bear \text{ arms, shall not be infringed.”}) = [1.04,\ 1.42, \cdots]$$

$$f(bear, \text{“A wild } bear \text{ was seen in Davis, CA.”}) = [2.14,\ 4.54, \cdots]$$

2.2.1 BERT

BERT [29] is a Transformer-based [85] natural language processing model from Google. It consists of multiple encoders stacked on top of each other. Each encoder takes a vector and encodes it into another vector which becomes the input to the next encoder. In contrast to

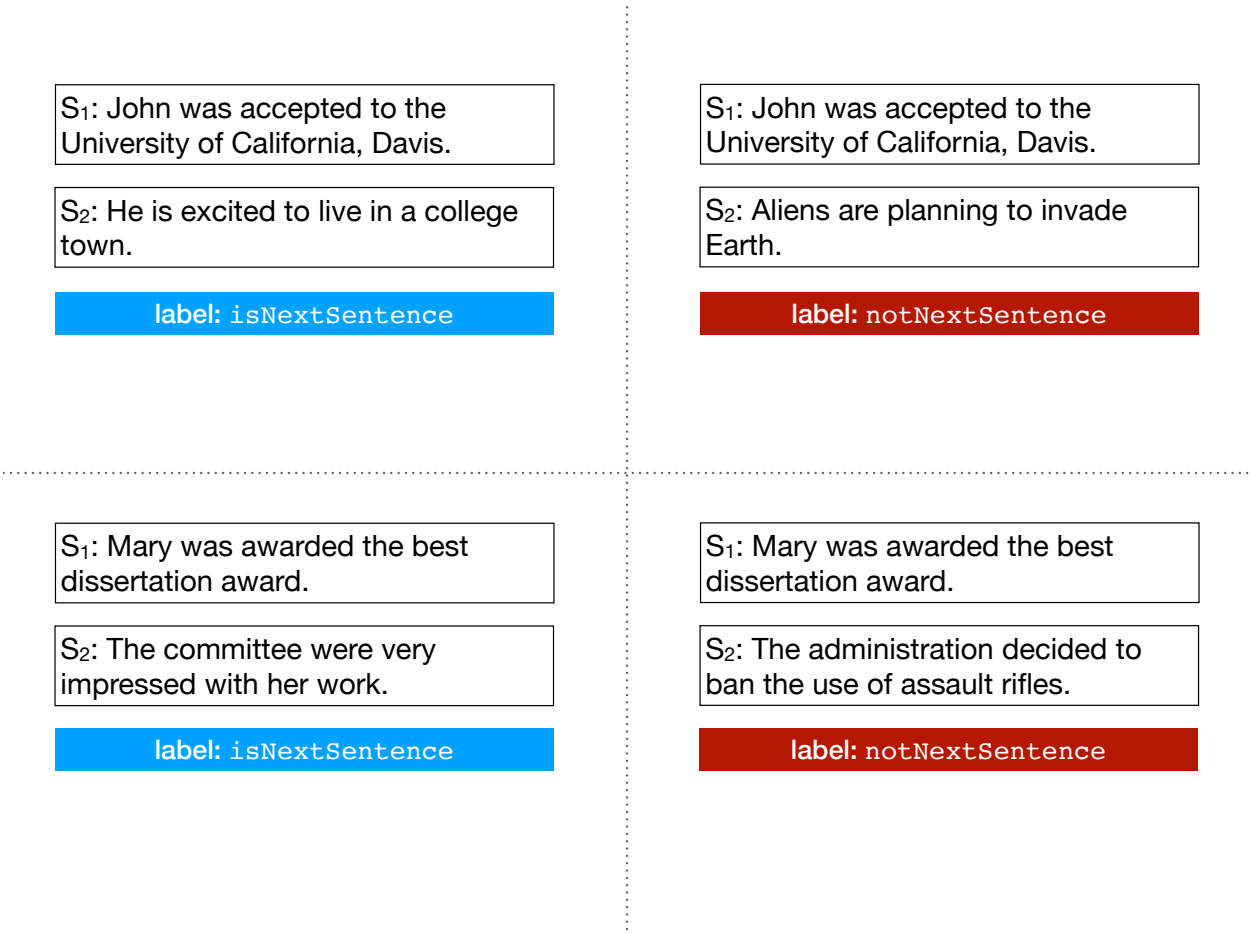

Figure 2.4: Examples of next sentence prediction task in BERT.

previous models that read the input sequence left-to-right or right-to-left, BERT reads the entire sequences of words at the same time. This allows it in theory to learn long distance relationships between words better. One potential problem with BERT is that words can see themselves in multi-layered context if we allow bidirectionality. The authors of BERT propose a solution to that: mask out a percentage on the input tokens and try to predict those masked tokens. In the BERT paper [29], the percentage is set to 15%, i.e. 15% of the tokens are masked and replaced with a special token (`[MASK]`) as shown in Figure 2.3. Then they train the model to predict the original tokens that were masked with a `[MASK]` token. They also train the model on next sentence prediction: given two sentences S_1 and S_2, predict whether S_2 is a sentence that proceeds sentence S_1 or not as shown in Figure 2.4. BERT is considered one of the main pillars of natural language processing and many variants

of it [53, 72, 52, 63, 90, 28, 14] exist today.

Pre-training and Fine-tuning

The terms pre-training and fine-tuning come up frequently when talking about BERT. We will explain the two terms and what they mean. Pre-training refers to the process of training a model from scratch on an unlabeled natural language corpus. The resulting model, called a pre-trained model, will include a generic knowledge about the natural language it was trained on. Fine-tuning refers to the task of adjusting a pre-trained model on a specific task (text classification, sequence tagging, etc.) given labeled data to benefit from what the pre-trained model has learned. For example, instead of training a sentiment analyzer that classifies English movie reviews to either positive or negative from scratch, we will utilize a pre-trained model that was trained on English Wikipedia to help us in training the classifier. Pre-training is usually much more expensive and slower than fine-tuning. In fact, the majority of people today do not have the adequate hardware requirements to pre-train a BERT model from scratch [77, 73].

While this pattern of pre-training and fine-tuning is the most common when it comes to BERT today, we can also generate vectors from BERT that can be used by other tasks or systems. One way to generate vectors is to return the last hidden layer. The BERT [29] paper goes into more detail about which layer to use when generating vectors.

Chapter 3

Learning from Sequences in Education: Representing Academic Degrees

3.1 Summary

In this work [4], we showcase the ability of word embedding models to generate representations of academic degrees from a sequence of courses. It is a common fact that a lot can be learned from sequences of words; however, it is not clear whether the same can be said for other sequences. In addition, we showcase how we can contract degrees using our representations and evaluate our approach on the majors and minors of the University of California, Berkeley.

3.2 Introduction

Both traditional and non-traditional educational institutions have been investing in automating many of their operations. One of the fundamental problems that arrive in these settings is how to create numerical representations of academic degrees, which are the essential of-

This chapter is based on work that was published at The 20th International Conference on Artificial Intelligence in Education (AIED 2019).

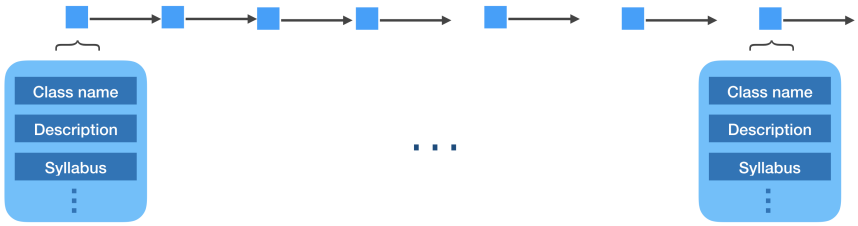

Figure 3.1: A sequence of courses.

ferings of these institutions. A degree usually consists of a set of courses and each course contains a name, a description, a syllabus, some learning outcomes, and much more. Modeling such complex entities is not straightforward, especially with all the different components involved. We propose a sequence driven method to learn such complex representations in academic settings without requiring any additional details.

3.3 Data

Before exploring the idea of learning degree representations from their sequences, we need a sequence of courses to use in learning the vectors as shown in Figure 3.1. The good news is that in academic settings we have a course sequence very similar to the one in Figure 3.1: student enrollments. All students, who graduated, have taken a sequence of courses that lead them to their degrees; we can learn from these sequences.

We use anonymized student enrollment data to train our embeddings. The dataset, provided by UC Berkeley, contains all student enrollments (over 140,000 students) from 2008-2015 across all departments and divisions. Table 3.1 shows the structure of the dataset. We preprocess the data by removing graduate students and filtering out graduate courses from undergraduates who have taken them, in order to focus the models on only the undergraduate curriculum. We also removed students who had been enrolled for less than eight semesters or more than twelve.

Table 3.1: The student enrollment dataset of the University of California, Berkeley

Student Masked ID	Year	Semester	Course ID
111	2010	Fall	Integrative Biology 127
222	2012	Spring	Mathematics 55

Student Masked ID	Major
111	Bioengineering
222	Computer Science

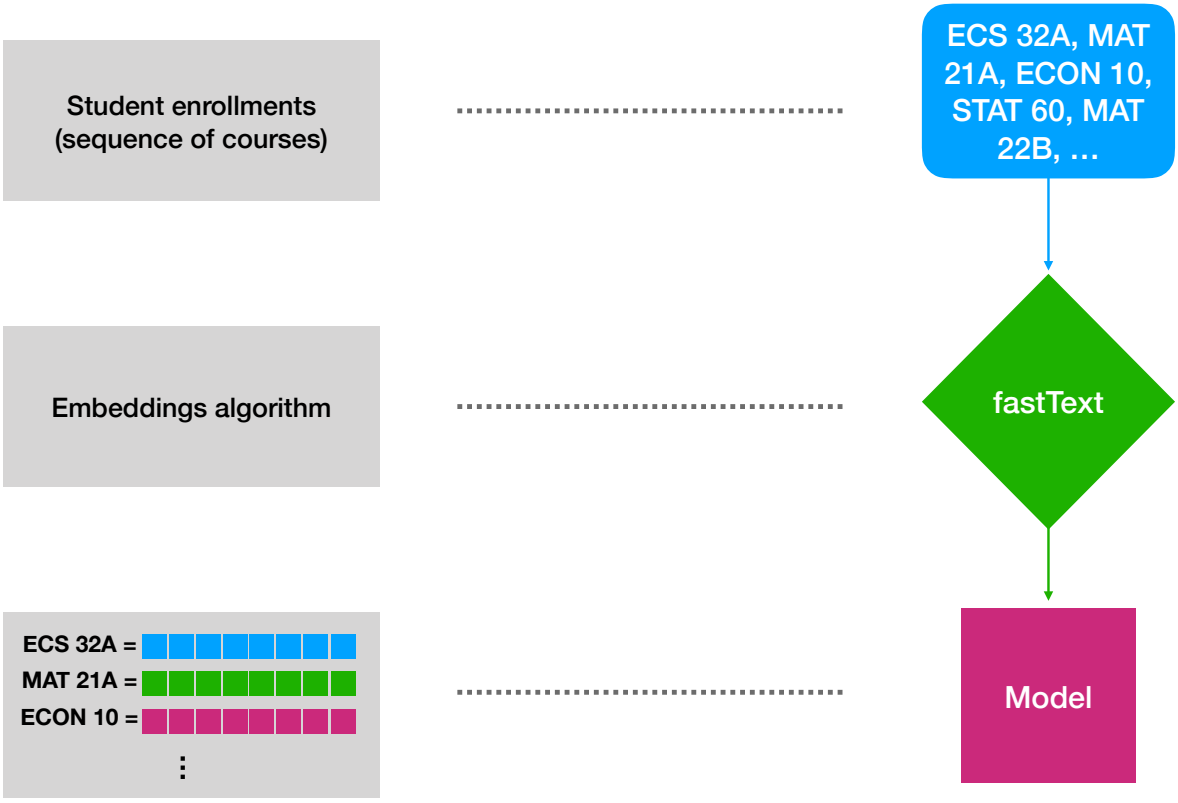

Figure 3.2: Applying word embeddings to course sequences.

3.4 Approach

Word embedding algorithms, such as word2vec [59], GloVe [65], and fastText [17], are powerful tools that allow us to represent a word by a high-dimensional vector while still capturing semantic information. To learn academic degree representations, we start by representing each student to be a sequence of the courses they've taken:

$$s_i = [c_{i_1}, c_{i_2}, c_{i_3}, \ldots, c_{i_n}] \tag{3.1}$$

where s_i is student i and $[c_{i_1}, \ldots, c_{i_n}]$ is the sequence of all the courses student i has taken. Notice that c_i only refers to the symbol of the class (e.g. Physics 7A) without any auxiliary information such as the course description and syllabus. Now we can train an embedding algorithm on these course sequences to get a vector representation for each course. Word embedding algorithms learn embeddings given a sequence of words; however, here we give them a sequence of courses which will result in course representations produced as shown in Figure 3.2. Now we can get a representation of any course:

$$course2vec(c_i) = [z_1, z_2, z_3, \ldots, z_n] \tag{3.2}$$

where c_i is course i and z_i's are real numbers learned by the word embedding algorithm. We can learn degree representations by averaging their course requirements as follow:

$$\frac{1}{|req(d)|} \sum_{c \in req(d)} course2vec(c) \tag{3.3}$$

where d is a degree and $req(d)$ is the degree requirements of degree d. There are two problems with such representation: 1) it requires the collection of the degree requirements which can be cumbersome if there are many requirements or degrees, and 2) the degree requirements are not defined clearly in some cases. To avoid that, we present a dynamic way to represent degrees as vectors learned from students' enrollments as follow:

$$degree2vec(d) = \frac{1}{|S_d||C_s|} \sum_{s \in S_d} \sum_{c \in C_s} course2vec(c) \tag{3.4}$$

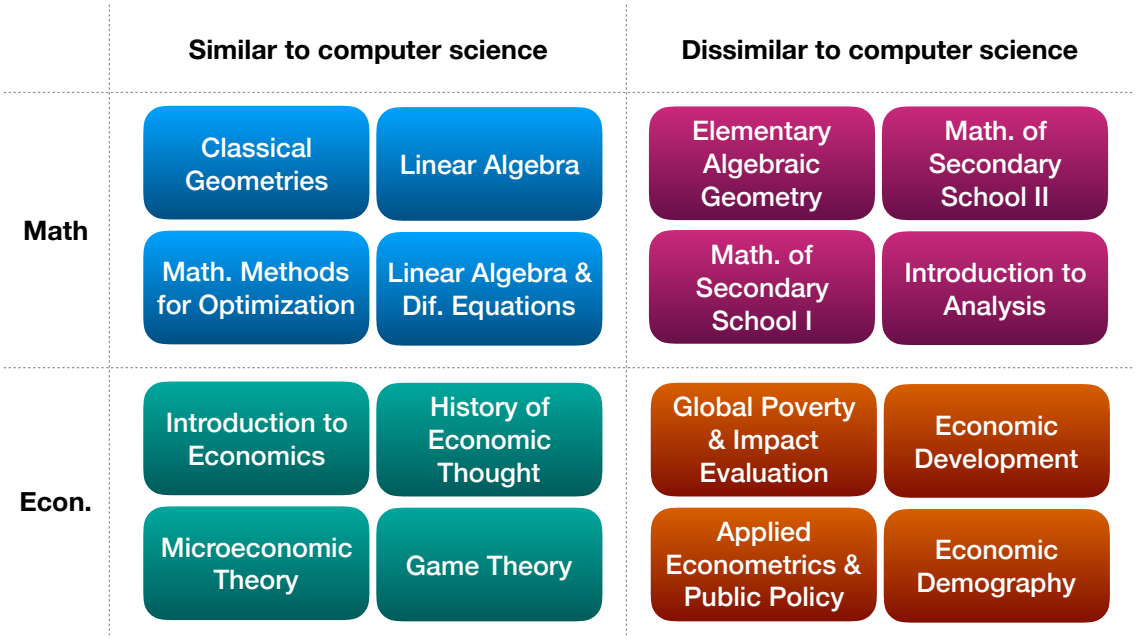

Figure 3.3: Finding math and economics courses that are similar/dissimilar to computer science.

where d is a degree, S_d is the set of all students majoring in d, and C_s is the set of all courses that student s has taken. This representation can be learned from the student enrollments without requiring additional information.

3.5 Experiments and Results

We trained a fastText [17] embedding model on the student enrollment dataset and computed the representations of the academic degrees of UC Berkeley. We noticed some interesting analogies in the representations such as:

$$\text{Mathematics} - \text{Computer Science} + \text{Mechanical Engineering} \approx \text{Physics} \tag{3.5}$$

which means that computer science has the same relationship to mathematics as mechanical engineering has to physics. Other interesting observations: *History of Art* and *Geology* are near the averages of *History* and *Art*; and *Geography* and *Geophysics*, respectively. Since we have representations for both degrees and courses, we can create a simple personalized

course recommendation system that can answer queries such as: "find me math courses that are similar to computer science." This can be translated to: "find math course vectors that are close to the computer science degree vector". The results of such query for example will include mathematical methods in optimization and linear algebra which are math classes that are indeed important for computer scientists. We can also ask the inverse: which math courses are very dissimilar from computer science. Our results show courses such as elementary algebraic geometry and mathematics of the secondary school curriculum as examples of answers to this query. Figure 3.3 shows the result of asking the model of the mathematics and economics courses that are similar to the degree of computer science, and also the ones that are not similar to computer science.

While we can see the relevance of these representations in the examples we provided, a more thorough evaluation should be made before reaching conclusions. Evaluating vector representations in relation to human needs is a complicated task since they are simply numerical representations that make sense to machines and not humans. One of the standard ways of evaluating them is to use them in downstream tasks that we, humans, can evaluate; e.g. we can use word embeddings to classify emails to spam and not spam and then evaluate the performance of the spam classifier. The task we use for evaluating our academic degree representations is curriculum contraction: we take an existing degree program, with a full roster of courses, and attempt to define a smaller set of courses that approximate it.

This task is becoming more important nowadays since educational platforms and institutions are looking to "right-size" the curricular experience of learners. This ranges from offering traditional four-year bachelor's degrees to six course "micro degree" credentials [38]. Curriculum contraction can be defined, in an academic context, as the process of contracting the length of a university degree program while retaining as much of the core value as possible (i.e., which courses should be chosen in a 1-year version of a 4-year program?). This problem does not arise only in traditional academic settings; but also in training and online course providers who are looking to "right-size" the curricular experience of learners. For instance, edX, the massive open online course provider, has introduced a program called

MicroMasters [38] that is like a contracted version of a typical master's program. One of the main selling points they use for the new program is that is faster and more flexible.

3.5.1 Curriculum Contraction

Let D be a degree plan consisting of courses. The high-level description of our contraction technique is as follow:

1. Embed the courses in a vector space.
2. Calculate a degree representation vector.
3. Find the best set of classes of size k that approximates the degree representation.

For steps one and two, we use *course2vec* and *degree2vec* respectively. For the third step, we want the best subset of courses that are closest to the degree D:

$$contract(D, k) = \arg \min_{d \in \mathcal{P}_k(D)} \sqrt{[degree2vec(D) - \frac{1}{|d|} \sum_{c \in d} course2vec(c)]^2} \tag{3.6}$$

where $\mathcal{P}_k(D)$ is all subsets of D that are of size k. Finding all subsets is computationally expensive and is not feasible for typical classes sizes; picking 10 classes from a 100 class will yield more than ten trillion sets! One way to make it faster is to use a *greedy* approach to find the closest k courses instead of finding the closest set of size k. While the greedy solution works well in some problems, in our problem it does not. Degrees often consist of a mixture of topics where their composition is close to the degree but each topic on its own may not; the composition of programming and math is close to computer science but each of these topics on its own may not be. So, we want to avoid using a greedy solution since we do not want to lose semantic relationships; two courses may be far from the degree vector, but their average may be closer than any other course vector. We make the assumption that although degrees consist of compositions, these compositions do not usually involve many components. As a result, we propose a *hybrid* approach where we do not explore all k-subsets but also do not lose compositional semantics. Instead of finding the closest set of

size k, we find the closest set of size of size four. After that, we remove the best four courses $\{x_1, x_2, x_3, x_4\}$ from D and repeat the process by updating the value of k to be $k-4$ and picking the best set of size four from $D - \{x_1, x_2, x_3, x_4\}$ and so on. Eventually, when k is small, we take combinations of sets of sizes: three, two, and/or one.

We use academic majors and minors to evaluate our contraction technique. Minors can be thought of as a compressed version of a major. In addition to that, they were carefully designed by educators who know a lot about their fields. We use majors and minors of the University of California, Berkeley to evaluate our representations: we take an academic major, contract it, and then compare it with its corresponding minor. A successful approximation should be very similar to the corresponding minor. We picked the ten most popular majors in UC Berkeley from 2010-11 to 2014-15 [64] that have corresponding minors to evaluate our approach. We apply *degree2vec* on each major m to get its embedding. After that, we create a set containing all department courses that students majoring in m have taken and remove courses from other departments. While some minors contain courses from other departments, we do not want to handpick these departments as it may add bias to the evaluation. We then run our contraction to get k courses. We pick k to be equal to the number of courses in the corresponding minor for each major. The recall@k is then measured for each major. Recall@k gives us the proportion of the minor classes we found in the contraction. Another way to think about recall@k is to view it as an answer to the question: if I take the classes proposed by the contraction, what percentage of the real minor would I cover? A recall@k value of 100% means that the contraction completely satisfies the minor, a recall@k value of 50% means that the contraction satisfies half the minor, and a recall@k value of 0% means that the contraction does not satisfy any minor requirements. Table 3.2 shows the performance of our representations in contracting majors. It is important to note that it is impossible for us to achieve perfect recall@k since we only take department courses; the maximum recall@k we can achieve is 87.88%. Figure 3.4 shows the t-SNE[1] [55]

[1]t-SNE is a popular visualization technique that is used to project high-dimensional spaces to two dimensions that still capture relationships found in those higher dimensions.

Table 3.2: Performance of our approach for each major.

Major	Recall@k
Electrical Engineering & Computer Sciences (EECS)	57.14%
History	83.33%
Mechanical Engineering	71.43%
Anthropology	75%
Architecture	37.5%
Chemical Engineering	40%
Statistics	44.44%
Rhetoric	57.14%
Environmental Economics & Policy	50%
Philosophy	66.67%
Average Recall@k	58.27%

projection of the resulting representations of all the majors and minors.

As we see in the results in Table 3.2, we are able to use these representations in performing a complicated task (contraction) with impressive results and show that these representations learned from courses sequences capture rich semantics.

3.6 Conclusion

We showed in this work that we can achieve good performance in curriculum contraction by using a simple vector space model that does not incorporate any textual information about the courses. There are many ways in which this work can be extended including a more sophisticated way of combining courses instead of averaging, and a more comprehensive technique to include possibly relevant courses from departments outside of the major by finding department vectors that are close to the main department vector.

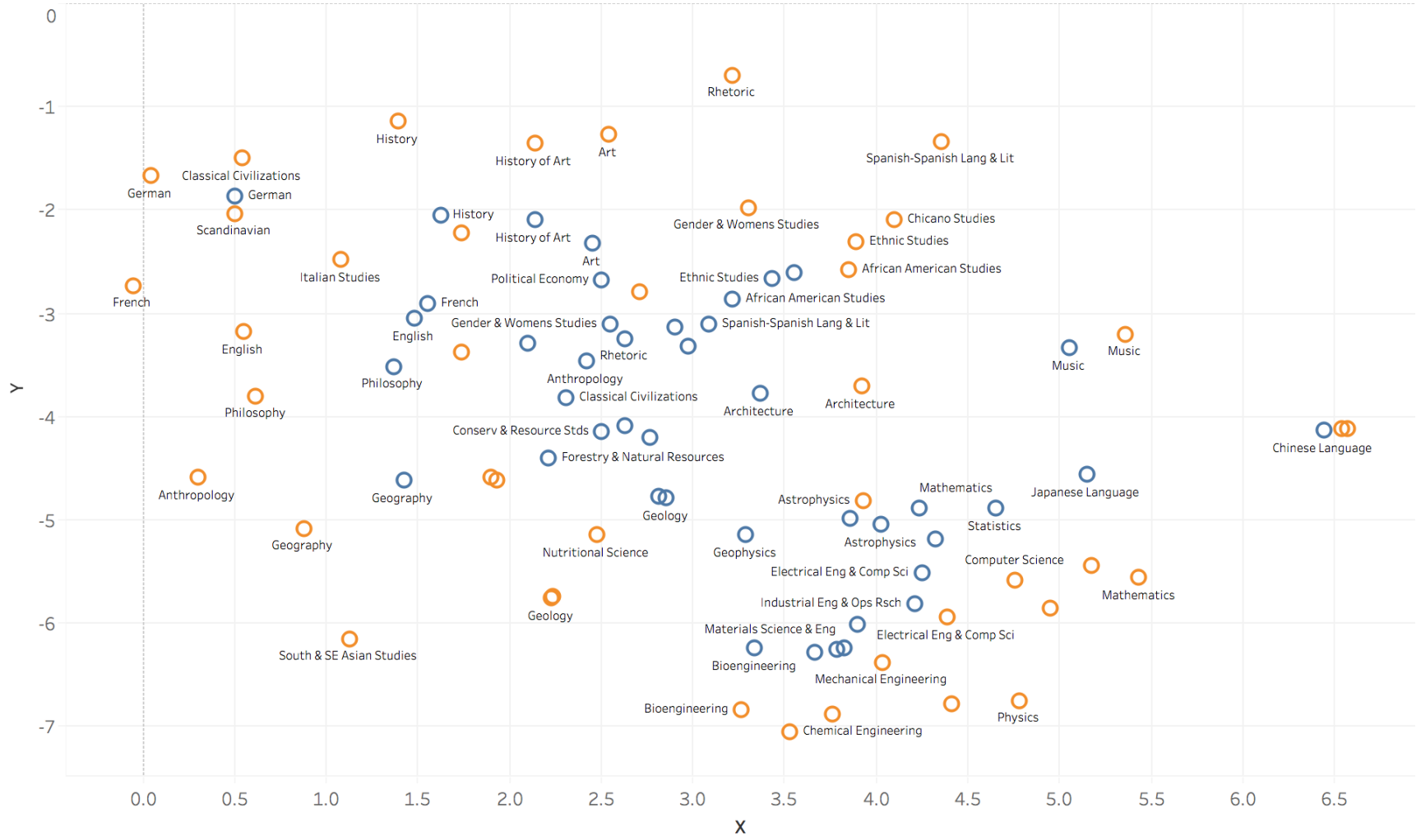

Figure 3.4: Visualization of the t-SNE projection of our embeddings for majors (blue) and minors (orange).

Chapter 4

Learning from Sequences in History: Identifying Narrators in Classical Arabic Texts

4.1 Summary

One widespread historical method of transmitting and recording information about important events and people in the Middle East is the narration-based method. In this method, each saying about a person or event is transmitted from person to person until a systematic collector records and compiles such sayings in a stable collection. At each stage of transmission, the narrator not only transmits the saying but also the person he got it from going back to the earliest narrator. Identifying each narrator in these collections is important to better measure the accuracy of the narrations and identify the date and geographies of the circulation of the sayings. In this work, we propose a natural language processing technique to automate the identification of narrators in classical Arabic texts. Our proposed technique consists of two models: 1) a model for detecting the narrators in the text, and 2) a model

This chapter contains work that was presented at The Islamicate Digital Humanities Network (IDHN) Conference on Digital Hadith Studies. Parts of this chapter are currently under review at The Arabic Computational Linguistics Conference (ACLing).

for linking narrators to their biographies. We train our two models on a large collection of annotated classical Arabic texts and achieve F1-scores of 96.15% and 95.74% for narration detection and linking respectively. Additionally, our model was able to find a mistake in the largest annotated classical Arabic texts corpus.

4.2 Introduction

There are many ways to record the sources of information transmitted about historical events and people. In the modern age, historians cite archival documents by referencing some combination of identifiers uniquely identifying them. Journalists indicate names of individuals or institutions as the source of a quote. During the 7th to 9th centuries, Muslim scholars developed and used a citation system that consisted of documenting the series of names signifying the transmission of a text from person to person before it was recorded by a systematic collector [22, 60]. This list of names, called the isnad, preceded each unique text or quotation found in the work of the systematic collector, and often took the following forms:

> It has been related to me by A on the authority of B on the authority of C that E said ...

and:

> A reported that: B narrated to us from C from D from E from F from G who said, 'I heard the H saying that J was next to K when he said ...'

The isnad-based system of citation was especially used to record the transmission history of reports about past events and peoples and remained a mainstay of scholarly citation practice for nearly a millennium. There exist hundreds of systematic works devoted to collecting these reports. Though, to our knowledge, no one has counted them, we conjecture that the number of reports in the systematic collections reaches in the hundreds of thousands, and that the number of unique individual narrators cited in them numbers in the tens

of thousands. These narrations serve as an important source for knowledge of early and medieval Islam.

One important subset of these reports are those that purport to go back to the Prophet Muhammad, called hadiths. In fact, some scholars have argued that the isnad-based system of citation first emerged in response to the desire to discriminate between authentic and fabricated reports about the Prophet and other important figures from the first generations of Islam. In addition to requiring that transmitters of hadith cite their sources, some scholars started collecting basic biographical information about the transmitters themselves, in addition to recording the judgments of hadith experts about the reliability and precision of their transmission practices. Eventually, some scholars of hadith created biographical dictionaries devoted to transmitters found in the isnads of hadiths. At a minimum, hadith literature consists of two basic types of works: those that contain hadiths with isnads, and those that contain information about the transmitters found in the isnads. Scholars of hadith use information in both of these sources to make judgements about when and where a text circulated and whether it can be authenticated to its purported source. However, the process of identifying narrators in the isnads and searching through the many volumes of biographical dictionaries is laborious; the names of the narrators cited in the isnads of the reports can be ambiguous; they can be first names, nicknames, a relationship (e.g. 'I heard my uncle say'), or something else. Moreover, the same narrator and the way they are mentioned in the text often differs from one text to the next. Manually finding biographical information on every narrator found in even a set of 10-20 hadiths can take many days. An example of a real narration is shown below:

> Aḥmad reported that: ʿAbd al-Raḥmān narrated to us from Mālik from al-Zuhrī from ʿAbbād ibn Tamīm from his uncle who said: I saw the Messenger of God ...

At a fundamental level, the isnad is simply a sequence of narrators that transmitted a text in serial fashion. While this structure is clearly important for dating and studying

the authenticity of historical texts, it has uses in other domains due to its rich structure. Since all narrators, except the first and last, are linked to two other narrators, a large transmission social network can be constructed from the narrators. This network will have multiple dimensions: spatial, cities where the narrators lived; temporal, the time when the narrator lived, etc.; resulting in a huge historical social network that would be valuable to network scientists and researchers. In addition, these transmitted sayings can be beneficial in understanding the evolution of language across geographical regions since we can date and know from which region the transmitters came. Furthermore, there are over thirty countries in the world that rely on isnad-based transmission to authenticate authoritative legal texts in their legal systems [26] which is why this issue is also of importance to legal scholars.

While the information contained in the isnad is clearly important, extracting that information is challenging because it is all documented in unstructured texts, and getting from the unstructured textual representation such as this:

> Ibn Abī ʿUmar told me that Sufyān told him on the authority of al-Aʿmash on the authority of Shaqīq on the authority of ʿAbdullāh: that the Messenger of God said: "Whenever there are three of you, then let two not converse to the exclusion of their companion."

to a structured construct where each narrator is detected and identified is no small problem. For example, there are over 20 narrators named Sufyān and over 200 narrators are called Ibn Abī ʿUmar. To our knowledge, no one has succeeded in automating the identification of narrators in isnads. In this work, we propose a method to do so that utilizes state-of-the-art natural language processing techniques. Our contributions can be summarized as:

- We develop a system that automatically identifies narrators in classical Arabic texts.
- We build ukhBERT: a BERT-based NLP model that is fine-tuned for hadith data.
- We propose a new way to tackle named-entity disambiguation (NED) problems in Arabic by posing them as token classification problems.

- We find a potential error in the largest annotated classical Arabic texts corpus.

The rest of the paper is structured as follows: Section 4.3 discusses related work; Section 4.4 defines the data collection process; Section 4.5 formulates the problem and our approach in solving it; Section 4.6 details the experiments and the results; and Section 4.7 concludes by recapitulating our findings and discussing potential directions for future work.

4.3 Related Work

There has been a growing interest in computational research in the area of hadith science recently [20, 12]. Muther and Smith [62] studied the problem of locating the start points of isnads in classical Arabic texts. Altammami et al. [8] used an n-grams model to segment hadiths into isnad and matn (body). Siddiqui et al. [61] used naive Bayes, decision trees, and k-nearest neighbors to segment hadiths. Azmi and Bin Badia [13] and Maraoui et al. [56] both worked on the problem of detecting isnads by using rule-based approaches. While our work extends this research, it differs in many ways to what has been done. First, the problems we tackle are much more complicated; we locate all narrators in the text and identify who each narrator is, a problem that no one has yet to solve. In addition, we rely on a dataset that is several times larger than data used in previous studies. Much previous work relied on one book (Ṣaḥīḥ al-Bukhārī). In contrast, our data consists of hadiths drawn from around 1,400 books. In fact, our testing dataset alone is larger than the combined data of all the previous work we mentioned.

4.4 Data Collection

Recently, digitized collections of both hadiths and biographical dictionaries of transmitters have emerged and made this part of historical research much easier. One such collection, Gawāmiʿal-Kalim (GK) [44], has not only digitized the largest collection of hadiths, but also hyperlinked the names found in isnads of hadiths to a table containing biographical information taken from the biographies of the narrators. They have done this for 447,205 hadiths, restricting themselves only to texts that purport to originate with the Prophet Muhammad,

Table 4.1: The structure of the hadith table.

ID	bookID	hadithID	hadithText
1	1	1	حدثني أبو عبيدة مسلم بن أبي كريمة التميمي جابر بن زيد الأزدي عن عبد الله بن عباس عن ...
1	1	2	أخبرنا عبد الرحمن قال ثنا إبراهيم قال نا آدم قال ثنا ورقاء عن ابن أبي نجيح عن مجاهد قال ...

even though they digitized 828,841 total reports, including those that do not go back to the Prophet. Thus, even within their own collection the narrators in isnads of 381,636 remain unlinked to their biographical information. Over 350 researchers and scholars worked on annotating the GK corpus and it contains, in total, around 1,400 sources: 900 are hadith collections and the rest are sources relating to Islamic law, theology, literature, etc. We relied on the data created by GK to create our dataset. One of the main downsides of the GK project is that the data is encoded in a custom format with no documentation. We spent months investigating that format and wrote multiple scripts to convert that format to a MySQL database with several tables. We also wrote scripts to extract the locations/indices of each narrator for every text giving us a mapping between every text and the IDs of its narrators. One of the most important tables amongst them is the biographies table which includes details of 48,937 narrators in addition to a special narratorID for unknown narrators. Table 4.2 shows the structure of the biographies table. The table contains information that was extracted from multiple sources and summarized into 30 columns. As we notice, in additional to all the information about each narrator, everyone is assigned a unique identification number (narratorID). Other important tables are the hadith and narrator linking tables which show the list of texts and a linking between narrators in the texts and their ID number, respectively. Tables 4.1 and 4.3 show the structure of the hadith and narrator linking tables.

Table 4.2: The structure of the biographies table.

narratorID	gender	name	teknonym (kunya)
1	male	محمد بن إدريس الشافعي	أبو عبدالله
2	female	هند بنت حذيفة	أم سلمة

narratorID	yearOfBirth	yearOfDeath	citiesLivedIn	...
1	150	204	Baghdad, Makkah, Egypt	
2	N/A	63	N/A	...

Table 4.3: The structure of the narrator linking table.

ID	bookID	hadithID	narratorAsMentionedInText	narratorID
1	1	1	أبو عبيدة مسلم بن أبي كريمة التميمي	31544

4.5 Approach

The problem of identifying narrators is a complicated problem even for humans: one first has to understand the text itself, find which parts of the text correspond to narrator names, find out possible narrators given their reported names, and then filter out the possibilities based on who's narrating from whom. We propose a natural language processing model that looks at the text, finds the locations of narrators in the text, and identifies each narrator by linking them to an entry in the biographies table. Our model is a BERT-based [29] model and we call it ukhBERT (أُخبرت), which means 'I was told' in Arabic. ukhBERT consists internally of two models that each solve an independent task. The two tasks solved by ukhBERT are:

1. Narrator detection (ND): find the narrators, as they are mentioned, in a given text.

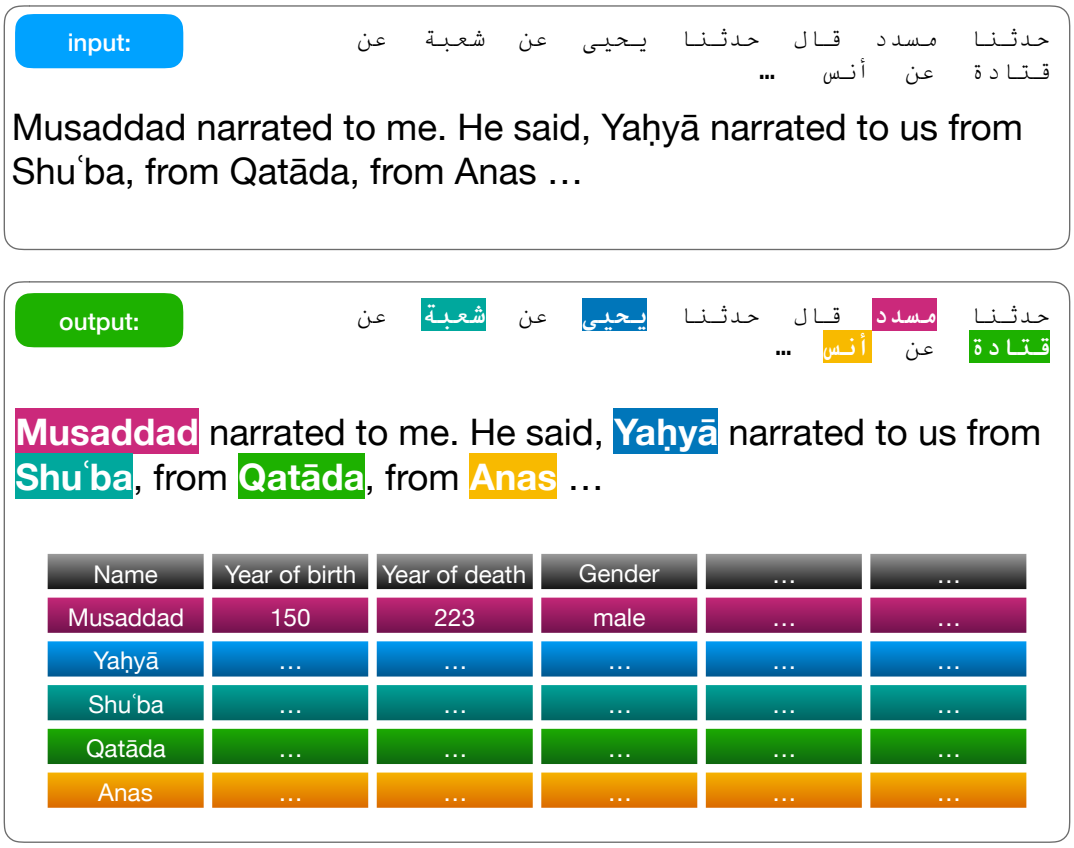

Figure 4.1: The input and output of the automatic narrator identification system.

2. Narrator linking (NL): identify the narrators given a sequence of narrators, i.e. link the narrators to their corresponding entries in the biography table discussed in section 4.4.

In the first part, the goal is to extract the names of the narrators as reported in the text as a list. In the narrator linking part, the goal is to take these names and relationships, who is narrating from whom, and find the most likely narrator to correspond to the reported name in the text. While both parts have their challenges, the latter problem is more difficult since it involves choosing between thousands of narrators for each name. Our system combines both steps to build an automatic narrator identification system that accepts raw text and then find all the narrators and identifies them as shown in Figure 4.1.

4.5.1 Narrator Detection

The goal of this task is to find the names of the narrators, as they appear in the text, given the text itself. Although the task seems similar in nature to a standard named-entity recognition (NER) problem, it is more complicated due to the long spans of entities/narrators and the fact that there are names in these texts that correspond to people but are not part of the isnad/narration sequence. To solve this task, we take the text and classify each of its tokens into one of three possible classes: `B-NAR` (beginning narrator), `I-NAR` (intermediate narrator) or `O` (other). These tags follow the IOB2 (Inside–Outside–Beginning) tagging format [68]. Let us take the previous isnad example we have seen in the introduction:

> *Aḥmad* reported that: *ʿAbd al-Raḥmān* narrated to us from *Mālik* from *al-Zuhrī* from *Abbād ibn Tamīm* from *his uncle* who said: "..."

Processing it using the proposed tagging format will result in the following:

> Aḥmad/`B-NAR` | reported/`O` | that/`O` | ʿAbd/`B-NAR` | al-Raḥmān/`I-NAR` | narrated/`O` | to/`O` | us/`O` | from/`O` | Mālik/`B-NAR` | from/`O` | al-Zuhrī/`B-NAR` | from/`O` | Abbād/`B-NAR` | ibn/`I-NAR` | Tamīm/`I-NAR` | from/`O` | his/`B-NAR` | uncle/`I-NAR` | who/`O` | said/`O`
>
> ...

After the tagging step, we create a list where each component consists of every sequence of tokens that begins with a `B-NAR` tag followed by zero or more `I-NAR` tags. That list contains the names (as reported in the text) of the narrators. The previous example will generate the following list: Aḥmad → ʿAbd al-Raḥmān → Mālik → al-Zuhrī → Abbād ibn Tamīm → his uncle. So how do we learn all of that? We use a BERT model [29] to learn this sequence tagging task. We use the large set of annotated texts discussed in section 4.4 to create training samples similar to the ones shown above. We then take a pretrained BERT model and fine-tune it on our task to be able to generate a narrator list given a text.

4.5.2 Narrator Linking

In this part we link each narrator in the list to one of ∼50,000 narrators we have in our biographies table. That is, given a sequence of narrators: Aḥmad → ʿAbd al-Raḥmān → Mālik → al-Zuhrī → Abbād ibn Tamīm → his uncle; we want to generate a narrator ID for each narrator. How can we do this? One way is to search the biographies by name and try to link the narrator to the corresponding row in the biography table. The problem is that this process is not clearly defined. Comparing names is not straightforward. Names can have multiple spellings. Furthermore, the spellings of the same names can change depending on their grammatical case, e.g. "Abū" can be written as "Abā" or "Abī" [1]. A further complexity is that narrators are cited many times by their relationships to other narrators ('his uncle', 'my father', etc.) instead of names or nicknames. Even if we successfully applied a set of predefined rules to account for these cases, we are not guaranteed to have a unique narrator in the end. This problem can be considered an entity linking (EL) or named-entity disambiguation (NED) task, which can be defined as the task of mapping some words of interest from a text document to a target knowledge base. The approach we propose utilizes the power of Transformer-based [85] models in learning these mappings by posing the problem as a sequence tagging problem. This approach does not rely on any predefined rules and just focuses on the sequential relationships between narrators, i.e. we treat the problem as a sequence tagging problem. This is similar to the approach we use in the narrator detection stage. However, we only had three possible outputs there (`B-NAR`, `I-NAR`, and `O`). Here we have tens of thousands of possible tags (narrators) which makes the problem more complex. For instance, given the following list: Aḥmad → ʿAbd al-Raḥmān → Mālik → al-Zuhrī → Abbād ibn Tamīm → his uncle; we will generate the following tags:

Aḥmad/`ID:488`	ʿAbd al-Raḥmān/`ID:4493`	Mālik/`ID:6659`	al-Zuhrī/`ID:7272`
Abbād_ibn_Tamīm/`ID:4130`	his_uncle/`ID:4818`		

This is the first work, to our knowledge, that shows the effectiveness of using Transformer-

[1] This phenomenon, the anomaly of narrators' names, is discussed in more detail by Azmi et al. [12].

Table 4.4: Performance of ukhBERT on narrator detection and linking.

Narrator detection	Precision	Recall	F1-score
ukhBERT (mBERT)	93.99%	98.25%	96.07%
ukhBERT (AraBERT)	**94.09%**	**98.30%**	**96.15%**
HMM	74.84%	70.50%	72.60%
CRF	81.52%	82.87%	82.19%
BiLSTM-CNN	89.84%	95.88%	92.76%
Narrator linking	Precision	Recall	F1-score
ukhBERT (mBERT)	**95.77%**	**95.71%**	**95.74%**
ukhBERT (AraBERT)	**95.77%**	**95.71%**	**95.74%**
BiLSTM-CNN	73.31%	75.59%	74.43%
Search-based	50.25%	47.80%	49.00%

Table 4.5: The size of the training and testing splits.

	Training set	Testing set
Number of books	899	499

based models in solving entity linking problems in Arabic by posing them as a token classification problem, which are simpler in nature and in implementation.

4.6 Results and Evaluation

We evaluated ukhBERT on the dataset discussed in section 4.4. We split the dataset into two splits: training and testing. To do so we start with the training set containing all books and the test set containing none; we then iterate through the books. For each book, we move it to the testing set if all the narrator IDs found in it are also found in the training set. We did so to ensure that the testing set contained no narrator ID that did not exist in the training set. In the end, the training set contained 899 books and the testing set consisted of 499 books as shown in Table 4.5. We intentionally split the set by books instead of taking a small sample from each book, because each individual work has a unique style of isnad

citation and organizational structure, and we want to evaluate the model on texts with styles that it had never seen before. As Siddiqui et al. [61] showed, a decrease in performance is observed when training on one book and then testing on another.

After splitting the set, we trained ukhBERT on the two tasks: narrator detection and linking by fine-tuning Google's pretrained multilingual BERT model (mBERT) [29] for each task for five epochs with the batch size set to eight to prevent memory issues on our GPU. We also fine-tuned AraBERTv0.1 [11] using the same hyperparameters. We picked AraBERTv0.1, instead of AraBERTv1, since it performs better on NER [11]. We compare our narrator detector to three models: 1) HMM [67], 2) CRF [51], and 3) BiLSTM-CNN [24, 54]. All three were trained on the same training data as our model. For the narrator linking problem we compare our model to a BiLSTM-CNN model, but we do not compare our narrator linking to the other sequence tagging solutions since they were not designed to be used with tens of thousands of tags. For instance, we estimated the time needed for an HMM model to predict our test set to be around nine years. We also compare our narrator linking model to a search-based name linking approach that we developed. The search-based approach relies on searching the entire biographies table, which contains extensive details about names for each narrator (full name, teknonym, nickname, etc.) spanning five columns, for names that match the narrator name as reported in the text. In the case where multiple narrators are found, we select the one that has narrated more texts. On the other hand, in case no narrators were found, we assign it the special ID that corresponds to unknown narrators. For both BiLSTM-CNN models, we use FastText's [17] pretrained Arabic word embeddings instead of random initialization. We set the batch size to 16, the learning rate to $1.5e^{-2}$ with a decay of 0.05, and dropout to 0.5. We train for 20 epochs and select the best performing one out of the 20. Both the HMM and CRF models were implemented using NLTK [16], while NCRF++ [89] was used for the BiLSTM-CNN implementation. Table 4.4 shows the performance of our models. We notice that ukhBERT performs better than competing approaches on both problems. What is more interesting is ukhBERT performance on the narrator linking task. That task is complicated even for humans, and one would expect that a model that

only takes into account sequential properties without looking at the knowledge base of the entities it's trying to link to perform poorly. Yet, ukhBERT achieves an F1-score of 95.74% on the narrator linking task beating both BiLSTM-CNN and the search-based approach by $> 20\%$ and $> 45\%$ respectively. We also notice in Table 4.4 the huge difference in the BiLSTM-CNN's performance on the two problems: from an F1 of 92.76% for narrator detection to 74.43% for narrator linking. It is interesting to note that using AraBERT instead of mBERT does not result in a huge increase in performance: in narrator detection we see a very small $(0.05 - 0.10\%)$ improvement while in narrator linking, we do not see any improvement. We think that fact that AraBERT was not pretrained on classical Arabic contributed to this.

4.6.1 The Automatic Identification of Narrators System

Figure 4.2 shows the interface of our automatic identification of narrators (AIN) system. The system is built with the Gradio [2] framework. Additionally, we use the spaCy [43] library to help in visualizing the tags. So why did we choose to name our system AIN (عين)? Due to following reasons: 1) it is an abbreviation formed from the initial letters of automatic identification of narrators, 2) it means 'eye' in Arabic which explains part of what the system is doing which is looking and investigating, and 3) it — AIN — also means an important person or figure in Arabic which also explains a main function of what the system is doing. In fact, a better question to ask is: why not name the system AIN?

4.6.2 Error Analysis

We investigated some of the error cases and will discuss two examples. We will try to understand why the model behaved incorrectly in those two cases. The first case concerns the following text:

> Muḥammad ibn Maslama told us that Yazīd ibn Hārūn said: Muḥammad ibn Isḥāq told me that ʿAbdullāh ibn Abī Najīḥ narrated to me on the authority of

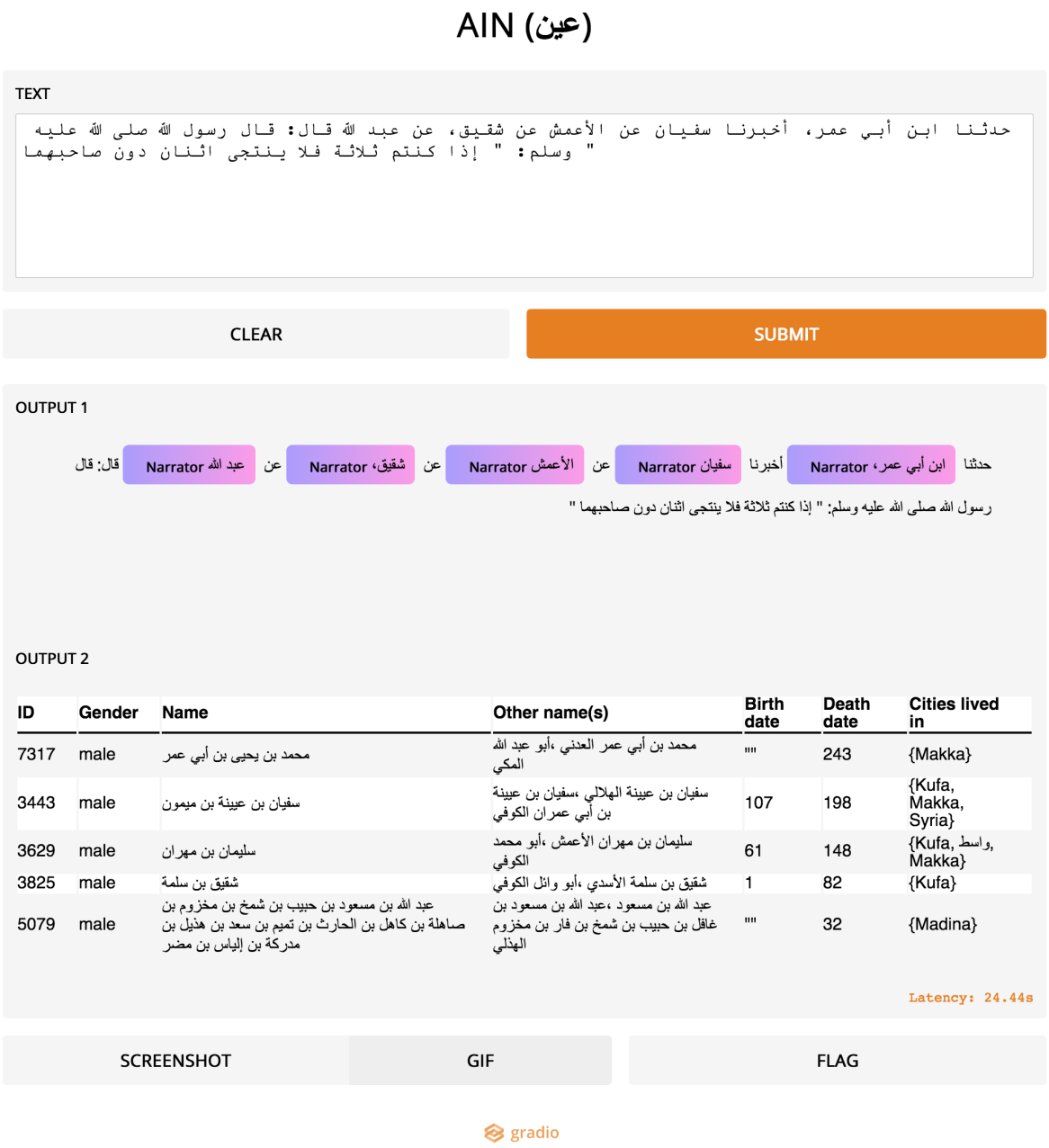

Figure 4.2: The interface of our automatic identification of narrators system.

Mujāhid on the authority of Ibn ʿAbbās who said: ...

ثنا محمد بن مسلمة ثنا يزيد بن هارون قال: قال محمد بن إسحاق حدثني عبد الله بن أبي نجيح عن مجاهد عن ابن عباس قال ...

In this example, the identifications produced by our system are the same as the ones from the golden set except in one narrator: *Muḥammad ibn Isḥāq*. In the labeled dataset this narrator is identified as *Muḥammad ibn Isḥāq al-Ṣāghānī* (ID: 6807). Our model identifies *Muḥammad ibn Isḥāq* as *Muḥammad ibn Isḥāq al-Yasār* (ID: 6811) as shown in Figure 4.3. At first look, it seems like our results are incorrect but upon further investigation we found something interesting: *Muḥammad ibn Isḥāq al-Ṣāghānī* died in 270 AH while *Muḥammad ibn Isḥāq al-Yasār* died in 150 AH. Now if we look at the narrator who Muḥammad ibn Isḥāq narrated from, Abdullāh ibn Abī Najīḥ, we will find that he died in 131 AH. It almost impossible for someone to narrate from a person that died 139 years before them. While we cannot say with 100% certainty who is the *Muḥammad ibn Isḥāq* mentioned in the text, we can say with confidence that our model's identification makes more sense than the original identification.

The second case concerns the narrator *ʿAlī* in the following text:

ʿImrān ibn Bakkār told us, he said: ʿAlī told us that ʿAbd al-Azīz ibn Ḥuṣayn Abū Sahl al-Khurasānī ...

ثنا عمران بن بكار، قال : حدثنا علي، ثنا عبد العزيز بن حصين أبو سهل الخراساني ...

The manual labels identify *ʿAlī* as *ʿAlī ibn ʿAyyāsh* (ID:5804) while our model identifies him as *ʿAlī ibn Mushir* (ID:5816) as shown in Figure 4.4. Both narrators lived in the same era; they died 210 AH and 189 AH respectively. There's no other instance of either narrating

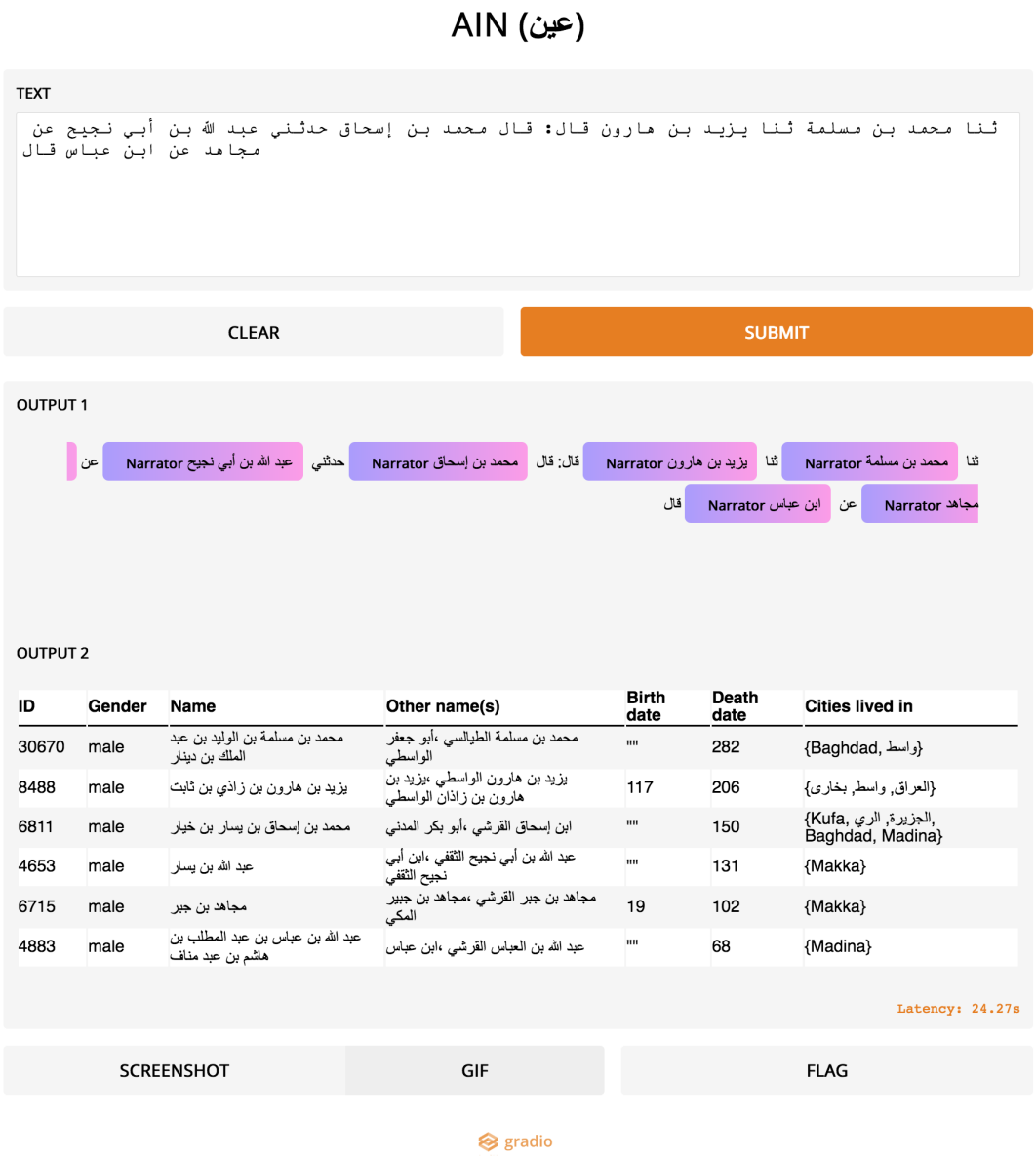

Figure 4.3: Our automatic identification of narrators system identifying Muḥammad ibn Isḥāq in the text: “Muḥammad ibn Maslama told us that Yazīd ibn Hārūn said: Muḥammad ibn Isḥāq told me that ʿAbdullāh ibn Abī Najīḥ narrated to me on the authority of Mujāhid on the authority of Ibn ʿAbbās who said: . . .” as Muḥammad ibn Isḥāq al-Yasār (ID: 6811).

from ʿAbd al-Azīz al-Khurasānī. *ʿAlī ibn ʿAyyāsh* occurs in 207 narrations, while *ʿAlī ibn Mushir* is more popular, occurring in 526 narrations. It seems that in this case, the model chose the the more popular option. When looking at the probabilities assigned to each by the model, we notice that *ʿAlī ibn ʿAyyāsh* is still ranked highly by the model (6th highest probability amongst all narrators).

4.6.3 Dealing with Special Cases

In this subsection we will discuss interesting examples and see how our system deals with them.

Dealing with relationships

We mentioned that sometimes a narrator is mentioned by their relationship to another narrator. We will experiment with different made-up examples of relationships and see how our system performs. The first two examples concern Ibn al-Zubayr (ID:4697), whose father and mother are also narrators. In the first example we will give the following input to our system:

> Ibn al-Zubayr told us on the authority of his father
>
> ثنا ابن الزبير عن أبيه

The system correctly identifies the father of Ibn al-Zubayr as Zubayr ibn al-ʿAwwām (ID:1427):

> Ibn al-Zubayr/`ID:4697` told us on the authority of his father/`ID:1427`

In the second example we will look at the following input:

> Ibn al-Zubayr told us on the authority of his mother

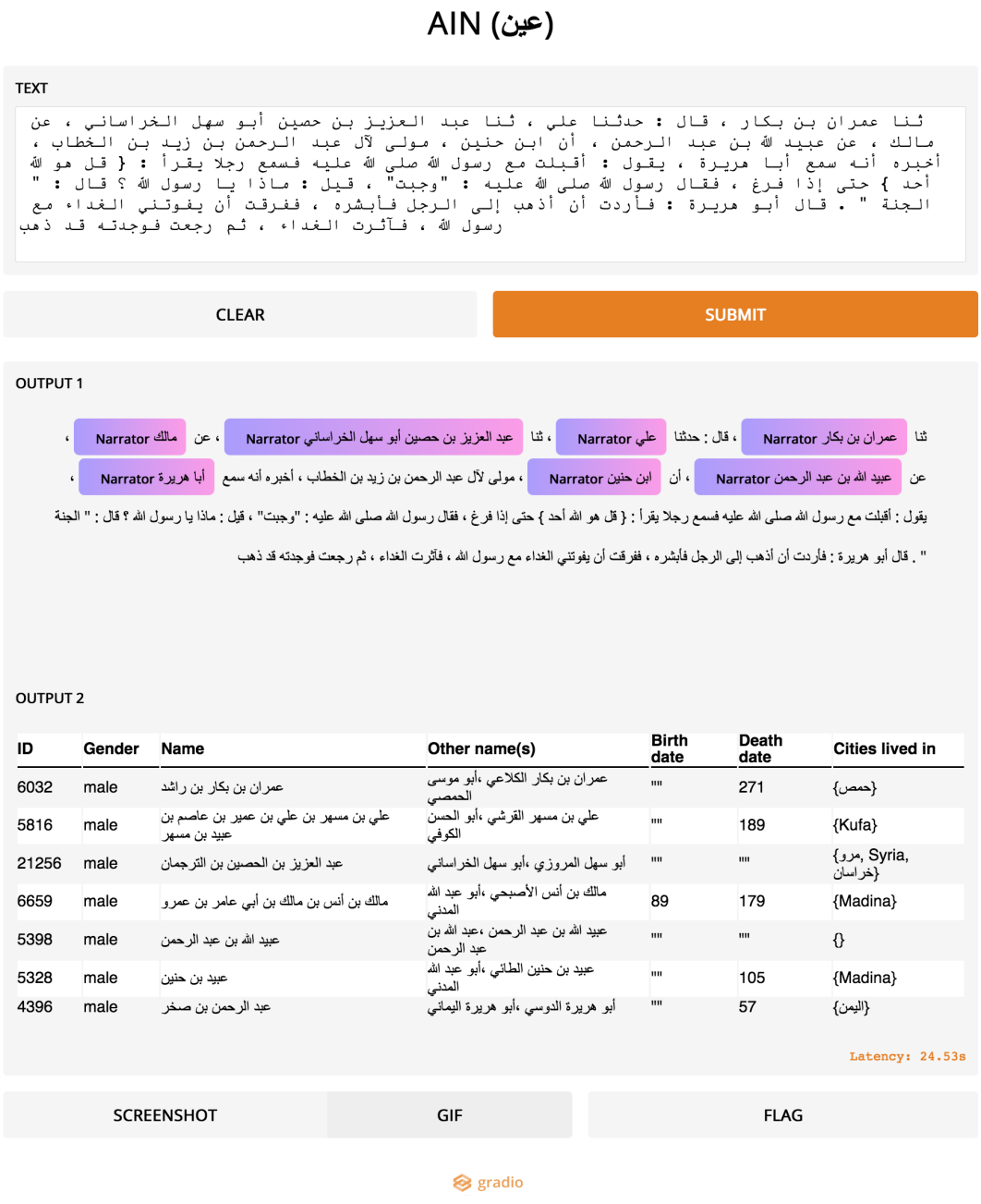

Figure 4.4: Our automatic identification of narrators system identifying Alī in the text: "Imrān ibn Bakkār told us, he said: ʿAlī told us that ʿAbd al-Azīz ibn Ḥuṣayn Abū Sahl al-Khurasānī . . . " as Alī ibn Mushir (ID:5816).

ثنا ابن الزبير عن أمه

This is a more interesting example since it is not common for people to narrate from their mothers. In fact, female narrators are disproportionately represented when it comes to narrators. In our database, female narrators represent $\sim 2\%$ of all narrators and participate only in 0.90% of all transmissions. Nonetheless, the system correctly identifies 'his mother' as Asmā' bint Abī Bakr (ID:553) who is Ibn al-Zubayr's mother:

> Ibn al-Zubayr/ID:4697 told us on the authority of his mother/ID:553

We decided to try more extreme examples where we have multiple consecutive relationships. Luckily, we have a narrator (Muḥammad al-Baqir, ID:7187) in our dataset whose father (ʿAlī Zayn al-ʿĀbidīn, ID:5739), grandfather (Husayn ibn ʿAli, ID:1336), and great-grandfather (ʿAlī ibn Abī Ṭālib, ID:5722) are all narrators. We tried the following example:

> Muḥammad al-Baqir told us that his father told him on the on the authority of his father on the authority of his father

ثنا محمد الباقر قال حدثني أبي عن أبيه عن أبيه

The correct identification in this case would be:

> Muḥammad al-Baqir/ID:7187 told us that his father*/ID:5739 told him on the on the authority of his father**/ID:1336 on the authority of his father***/ID:5722

Our system does not produce the correct result, instead it outputs the following identifications:

Muḥammad al-Baqir/ID:7187 told us that his father*/ID:7187 told him on the on the authority of his father**/ID:5739 on the authority of his father***/ID:1336

While at first glance we see that all the identifications of 'his father' are wrong, a deeper investigation reveals that is not exactly the case. The system associated the first instance of 'his father' (his father*) with Muḥammad al-Baqir himself. On the one hand, this is odd since narrators do not narrate from themselves. On the other hand, mistaking a father for his son is better than mistaking him for someone who is not related to him whatsoever. If we take into account that misidentification of his father*, the other identifications are all correct if we shift them; his father**/ID:5739 is indeed the parent of his father*/ID:7187, and his father***/ID:1336 is indeed the father of his his father**/ID:5739.

Dealing with animals

What happens when we replace a narrator name with an animal? To answer this question, we gave our system the three inputs shown in Table 4.6. As we notice, in two of the examples, 1 and 3, the animal is completely neglected. In the second example, the system identified lion as the narrator with ID:541. If we look at the information of that narrator, we find that his name is Asad ibn Mūsā. Since the word for lion[2] in Arabic is 'Asad', the system thought we were talking about a person named 'Asad' and not a lion.

Dealing with narrators that do not exist

We saw in the previous examples that replacing a narrator name with a name of an animal will result in the system neglecting that name. What if we replace a narrator name with a narrator that does not exist? What happens if we give the system a European name? While there are no European narrators in our dataset, a human who reads a narration that contains a European name will understand that it refers to a person even if they do not know who that person is. We constructed similar examples to the ones we saw in Table 4.6

[2]Since there are no indefinite articles (a/an) in Arabic; 'lion' and 'a lion' are written in the same way.

Table 4.6: Examples of texts where the narrator name is replaced with an animal.

Example 1	
	ثنا محمد عن كلب
Input	Muḥammad told us on the authority of a dog
Output	Muḥammad/ID:7016 told us on the authority of a dog
Example 2	
	ثنا محمد عن أسد
Input	Muḥammad told us on the authority of a lion
Output	Muḥammad/ID:6864 told us on the authority of a lion/ID:541
Example 3	
	ثنا محمد عن جمل
Input	Muḥammad told us on the authority of a camel
Output	Muḥammad/ID:7016 told us on the authority of a camel

to showcase the system behavior when encountering narrators with names that are unusual. We created four examples with the following names: Jessica, Elizabeth, George, and John. The system's output when given these names is shown in Table 4.7. As shown in Table 4.7, the system identifies the European narrators but labels them all with the ID:1131. What does that mean? The ID:1131 is a special narrator ID that the system uses for unknown narrators. Hence, the system is saying that it recognizes these people as narrators, but it is not confident in linking it them to any of the narrators it already knows. Another interesting note is how the system identified Muḥammad differently in each of the four examples. We are not sure why the system decided that a Muḥammad who narrated from George will be a different person than a Muḥammad narrating from Elizabeth. We tried replacing the European names with ambiguous references to people: woman, girl, man, and boy; and the system identified them all as unknown narrators with ID:1131 as shown in Table 4.8.

Table 4.7: Examples of texts where the narrator name is replaced with an Western name.

Example 1	
	ثنا محمد عن جيسيكا
Input	Muḥammad told us on the authority of Jessica
Output	Muḥammad/ID:6764 told us on the authority of Jessica/ID:1131
Example 2	
	ثنا محمد عن إليزابيث
Input	Muḥammad told us on the authority of Elizabeth
Output	Muḥammad/ID:6864 told us on the authority of Elizabeth/ID:1131
Example 1	
	ثنا محمد عن جورج
Input	Muḥammad told us on the authority of George
Output	Muḥammad/ID:6811 told us on the authority of George/ID:1131
Example 1	
	ثنا محمد عن جون
Input	Muḥammad told us on the authority of John
Output	Muḥammad/ID:7016 told us on the authority of John/ID:1131

4.7 Conclusion

In this work, we built an automatic narrator identification system that can link narrators in a given text to their biographies. We also showed the effectiveness of Transformers in identifying narrators; we achieved impressive performance even though we discard all information contained in the biographies and treat the problem of linking narrators as a sequence tagging problem. There are many ways in which this work can be extended including: training a classical Arabic BERT model instead of using a generic pretrained model and incorporating some of the biographical details into the model and see if that improves the performance.

Table 4.8: Examples of texts where the narrator name is replaced with: woman, girl, man, and boy.

Example 1	
	ثنا محمد عن امرأة
Input	Muḥammad told us on the authority of a woman
Output	Muḥammad/ID:7016 told us on the authority of a woman/ID:1131
Example 2	
	ثنا محمد عن بنت
Input	Muḥammad told us on the authority of a girl
Output	Muḥammad/ID:6864 told us on the authority of a girl/ID:1131
Example 1	
	ثنا محمد عن رجل
Input	Muḥammad told us on the authority of a man
Output	Muḥammad/ID:7016 told us on the authority of a man/ID:1131
Example 1	
	ثنا محمد عن ولد
Input	Muḥammad told us on the authority of a boy
Output	Muḥammad/ID: 6864 told us on the authority of a boy/ID:1131

Chapter 5

Learning from Sequences in Natural Language: Incorporating Morphology in Traditional Word Embeddings

5.1 Summary

In this work [5], we propose a new Arabic word embedding technique that produces smaller models than traditional methods by utilizing the complex morphology of Arabic. Our approach relies on segmenting words into subwords during training time and then composing word-level representations from subwords during test time. We train our embeddings on Arabic Wikipedia and show that they perform well in the Arabic word analogies dataset and other Arabic natural language processing tasks while being around 60% smaller. Moreover, we showcase our embeddings' ability to produce accurate representations of some out-of-vocabulary words that were not encountered before.

This chapter is based on work that was published at The Fifth Arabic Natural Language Processing Workshop (WANLP 2020) in The 28th International Conference on Computational Linguistics (COLING 2020).

5.2 Introduction

Words embeddings today are one of the main building blocks of most natural language processing tasks. From building text classifiers to creating named entity recognizers, having a robust word representation is an essential step. Many word embedding techniques have been proposed [59, 65, 17] that try to capture better word representation. Although most of the approaches are language agnostic, they are mainly designed to be used with English (and languages similar to it). Nonetheless, it was shown that these techniques work well in other languages [39, 79].

One of the main problems with word embedding models is their size spanning hundreds of megabytes and up to gigabytes. This is a bigger problem for the Arab world due to the limited broadband internet access availability and the widespread use of phones and tablets which are less capable than personal computers [45]. The advent of deep natural language processing in many fields calls for production-ready models that are smaller but still perform well.

In this work, we propose a new Arabic embedding technique that is smaller than those produced by the standard techniques by utilizing the morphologically complex nature of Arabic. Figure 5.1 contains a summary of our techniques and shows what happens in the training and inferences stages.

The rest of the chapter is organized as follows: Section 5.3 defines the process of gathering and cleaning our data; Section 5.4 highlights the embedding approach we are proposing; Section 5.5 details the experiments and the results; Section 5.6 discusses related work; and Section 5.7 concludes by summarizing the work.

5.3 Data

In this section, we describe the process of gathering and preparing our data. We use Arabic Wikipedia as a corpus. We downloaded the Wikipedia dump from January 2018 and then cleaned it by using WikiExtractor[1] which is a utility that generates plain text given a

[1]https://github.com/attardi/wikiextractor

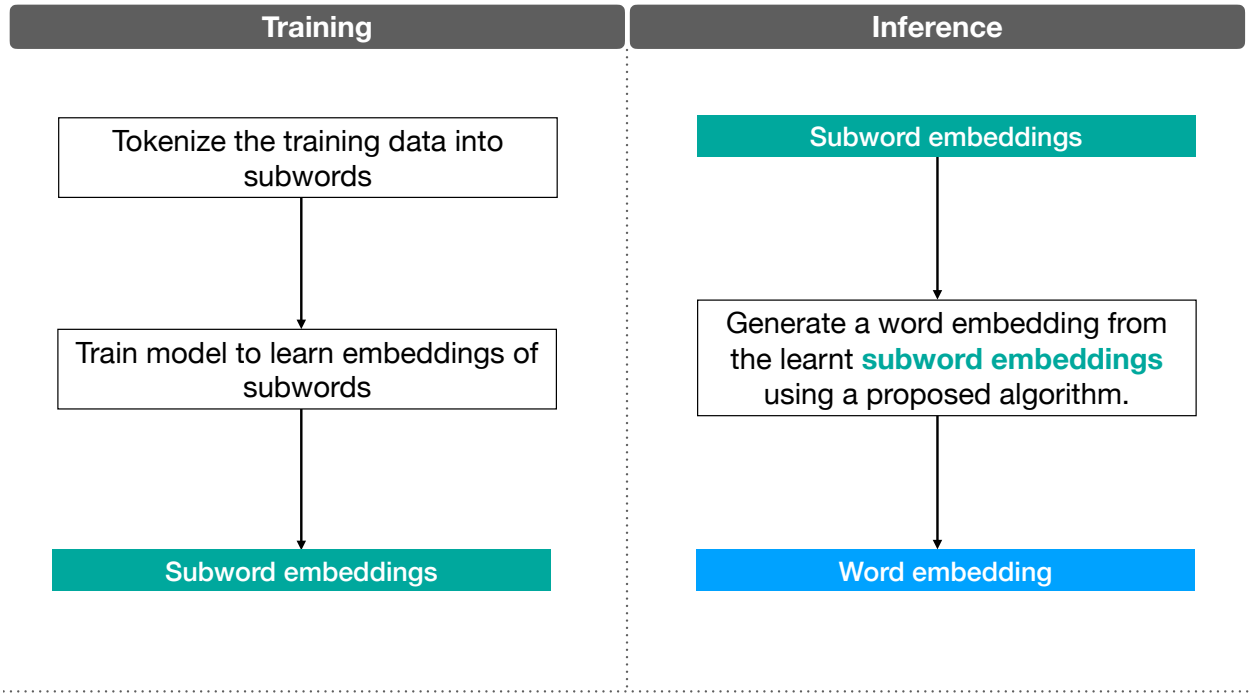

Figure 5.1: The training and inference stages of our proposed technique.

Wikipedia dump which is usually in XML [21]. After that, we use a custom set of regexes to filter out all non-Arabic words, such as English words and numbers, and remove all diacritics resulting in over 86 million tokens.

5.4 Approach

One example of Arabic's morphological complexity is in the number of verb forms it possesses which is much higher than in English as we can see in Table 5.1. In fact, Arabic has over a thousand possible verb forms: 13 person/number/gender forms times 9 tense/mood combinations times 17 form/voice combinations. Figure 5.3 shows an example of an Arabic word with its translation in English and its segmented parts. As we see, morphology makes words much more complex to represent.

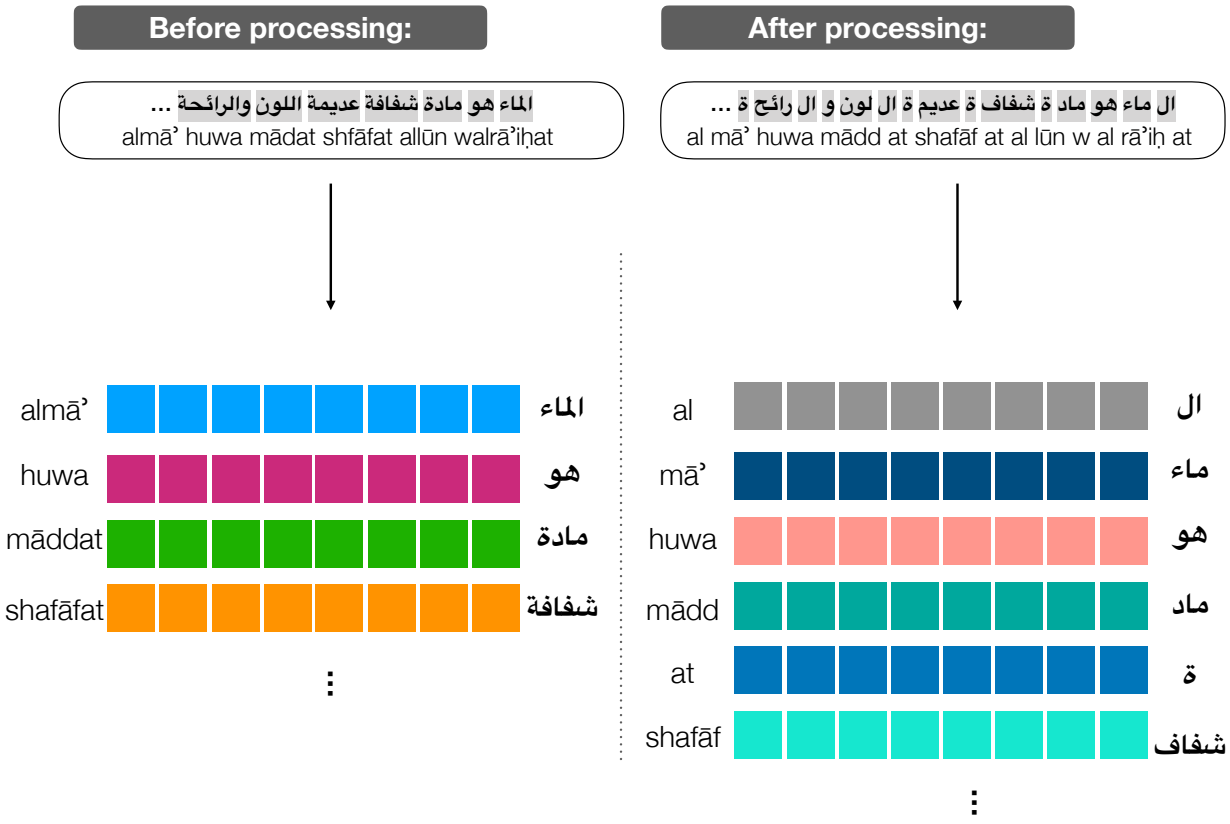

Figure 5.2: The effect of word segmentation on the resulting vectors. In the top we have representations of words and in the bottom, we have representations of subwords.

Figure 5.3: Decomposition of an Arabic word and all the parts it contains.

Verb	Forms
go	go, went, going, gone, goes
ذهب	ذهب (he went) , يذهب (he goes) , سيذهب (he will go),
	ذهبا (they [masculine dual] went), يذهبا (they [masculine dual] go),
	سيذهبا (they [masculine dual] will go), ذهبوا (they [plural] went),
	يذهبوا (they [plural] go), سيذهبوا (they [plural] will go),
	ذهبنا (we went) , نذهب (we go) , سنذهب (we will go),
	ذهبت (she went) , تذهب (she goes) , ستذهب (she will go),
	ذهبتا (they [feminine dual] went), تذهبا (they [feminine dual] go),
	ستذهبا (they [feminine dual] will go) , ⋯

Table 5.1: Verb forms in English and Arabic.

We believe that one of the reasons of the degraded representations in morphologically rich languages is because of how they are modeled as shown in Figure 5.4. While traditionally, this aspect of Arabic, and other morphologically rich languages, has been challenging to the natural language processing and computational linguistics communities [34, 3], we asked whether we may benefit from this characteristic.

Having a rich morphology can allow us to reduce a size of our vocabulary, which is the main bottleneck in word embedding models. It allows us to decompose words at a more granular level, which results in smaller models.

Word embedding models are trained on a large corpus of text. The main difference in our approach is that we preprocess the text before feeding it into the embedding algorithm by splitting every word into subwords, which are its prefix(es), stem, and suffix(es) using Farasa, an Arabic segmenter [1]. The effect of using the Farasa segmenter can be seen in Figure 5.2, where each row of squares represents a vector. Then we train the resulting corpus using Word2Vec, though we note that our approach is embedding agnostic and may be used with any embedding model. Notice that our vocabulary, and the resulting vectors, will be completely different now as shown in Figure 5.2. It may seem at first glance that our

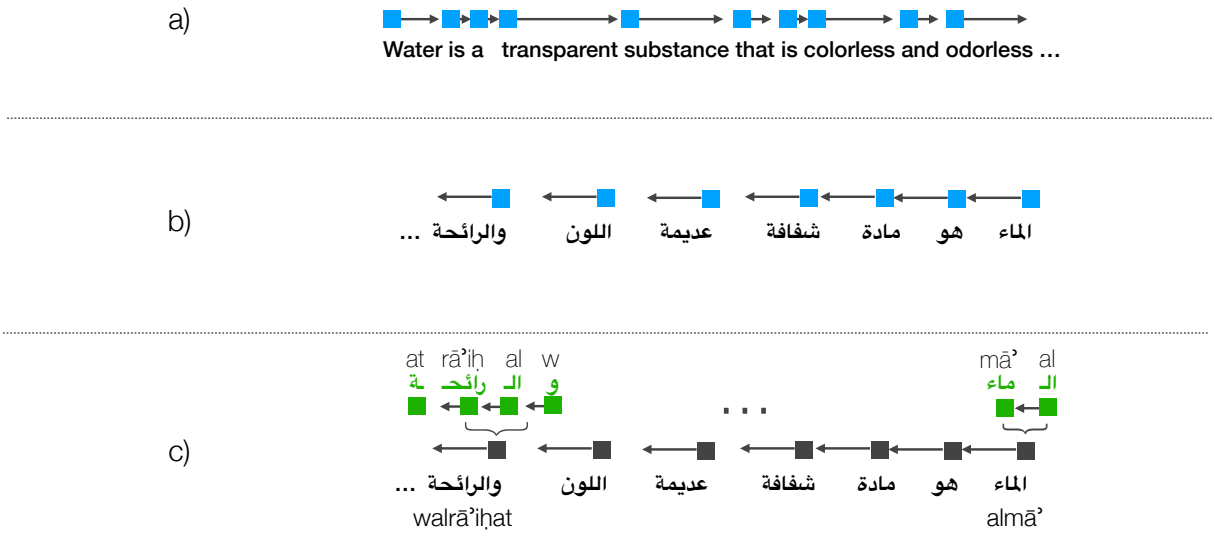

Figure 5.4: a) How English is usually modeled. b) How Arabic is usually modelled (similar to English) in existing embedding techniques. c) What we be believe to be a more accurate modelling of Arabic.

vocabulary is increasing, but in fact it decreases as we keep adding more words as shown in Figure 5.5, which depicts the sizes of the vocabularies in the first million words in Arabic Wikipedia.

One question that comes to mind is how do we generate embeddings for words that were split into subwords because in most cases we want embeddings at the word level and not at the subword level. We propose the following technique for getting the embeddings of all types of words, including those with multiple subwords. For words with only one component, we just return the embeddings learned by the model. For words with multiple components (subwords), we get the embedding of the longest subword, and the average of the embeddings of the remaining subwords. We then take a weighted average of the two values which results

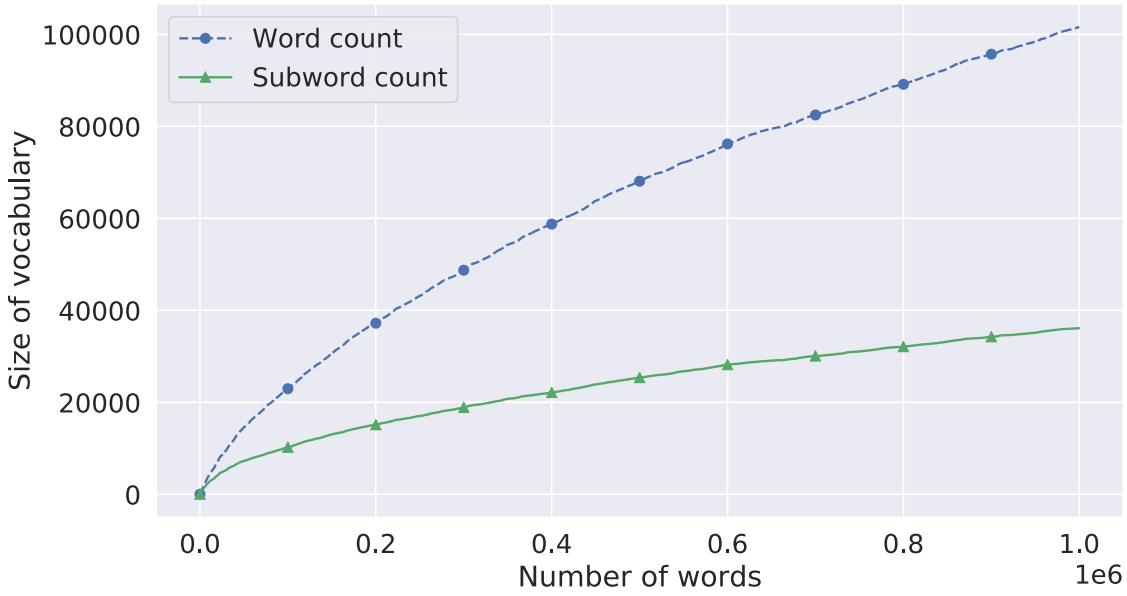

Figure 5.5: The increase in the size of the vocabulary when using words and subwords. The x-axis shows the number of words processed in the Arabic Wikipedia. The y-axis indicates the size of the vocabulary.

in our embedding as Equations 5.1 and 5.2 show.

$$l = \arg\max_{x \in S}(|x|) \tag{5.1}$$

$$v = \alpha * (M[l]) + (1 - \alpha) * (\sum_{x \in S \setminus \{l\}} \frac{M[x]}{n-1}) \tag{5.2}$$

where S is the set containing all the subwords contained in the word, M is the learnt model that contains embeddings for all subwords, and α is a parameter that decides the weights between the longest subword and the other subwords. Algorithm 1 illustrates the algorithm used in determining embeddings for all cases.

This method not only allows us to generate embeddings for all words in the original corpus, but also increases its capacity to deal with out-of-vocabulary (OOV) words that the model has never seen before as shown in Figure 5.6. For example, in Figure 5.6, we see how our model can produce a representation of 'and their iPhone' which is one word in Arabic.

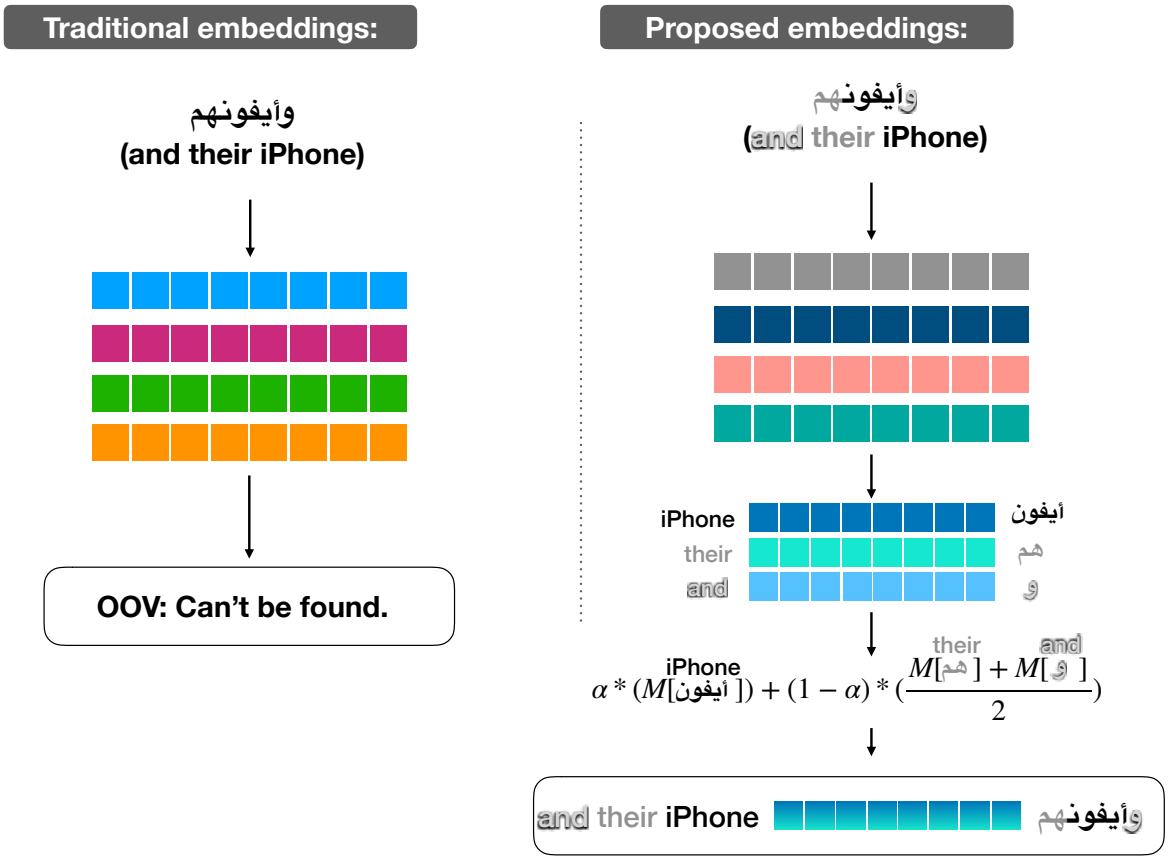

Figure 5.6: Dealing with out-of-vocabulary words in both the traditional models and our proposed model.

A traditional model trained on the Arabic Wikipedia will fail to produce a representation of the word 'and their iPhone' because that word never appeared in Wikipedia. In fact, since there are many forms for each word, no matter how large the training corpus is, it is almost impossible for it to have seen occurrences of all possible forms of all words in it. Our model can tackle this problem because it operates on a subword level and has seen all the three components that make up the word: 'and', 'their' and 'iPhone' as shown in Figure 5.6. This procedure allows one to expand a model's vocabulary without retraining or requiring numerous examples of a given word. Of course, not all out-of-vocabulary words will be found this way. Nonetheless, it's an efficient way to generate representations of new words that is not possible in classical approaches.

Algorithm 1 Generating embeddings of words from subwords

function GETEMBEDDING($word, \alpha, model$)

  if $word \in model$ **then**

    return $model[word]$

  else

    $S =$ get_components($word$)

    for $s \in S$ **do**

      if $s \notin model$ **then**

        return $error$

    $l = argmax_{s \in S}(|x|)$

    $S = S - l$

    return $\alpha * (model[l]) + (1 - \alpha) * (sum([model[s] \text{ for } s \in S]) \div |S|)$

5.5 Experiments and Results

5.5.1 Evaluation Datasets

For intrinsic evaluation we used the Arabic word analogy benchmark created by Elrazzaz et al. [33]. This dataset consists of nine relations that each consist of over 100 word pairs. We use the following datasets to evaluate our model extrinsically:

1. **APMD**: The Arab poem meters dataset [10] which consists of 55,440 poem verses with each verse classified into one of the fourteen Arabic poetry meters. The data is split into training and testing sets.

2. **HARD**: The Hotel Arabic Reviews Dataset [32] consists of 93,700 hotel reviews that are classified into positive or negative according to their rating. Reviews with a rating of four or five were assigned positive, and those with a rating of one or two were labeled negative. Reviews with a rating of three were ignored. We split the data into 80% and 20% training and testing sets respectively using the script provided by Antoun et al.[11].

3. **LABR**: The Large-scale Arabic Book Reviews dataset [9] consists of 63,000 book reviews rated between one and five. We use the unbalanced two-class dataset, where reviews with a rating of one or two are labeled negative, and those with a rating on four or five are labeled positive. The data is split into training and testing sets.

5.5.2 Intrinsic Evaluation

Word analogies, which consists of sets of pairs that share a common relationship, are often used to evaluate different embedding techniques. For example, let's say that we have the following pairs: (king, queen), and (male, female). Both of these pairs contain a word that indicates masculinity and a second word that indicates the feminine version of the first word. A perfect word embedding representation should be able to capture this relationship, and the way to mathematically measure it is by calculating the vector king−male+female. After that, we check to see whether the resulting vector is the closest to the vector queen or not. We use the Arabic word analogy benchmark created by Elrazzaz et al. [33] to evaluate our approach. We used Gensim's [69] word analogy evaluate function to evaluate the models and set the 'dummy4unknown' flag on so that all tuples of pairs that contain a word (or more) that are not in our vocabulary will get zero accuracy (instead of being skipped), similar to the procedure adopted by Elrazzaz et al. [33].

We train two models: a vanilla Word2Vec model (base model) and our proposed model on the Arabic Wikipedia dataset mentioned in Section 5.3. For the base model, we set the window size to be five. Since our model segments words before learning embeddings, we compute the average number of subwords per word in our corpus and adjust the window size by that number. The average number of components per word is around two (1.97); to account for that we set window size in our model to be ten instead of five. For both, we set the number of epochs to be equal to ten. We experimented with multiple α values for our proposed model and found that setting it to 0.3 achieves the best results. Moreover, to avoid vocabulary size discrepancies, we standardize the vocabulary size before the evaluation by ensuring that our model has the same vocabulary as the base model. To do that, we create a

Vector dimensions		dim=50	dim=100	dim=200
Skipgram	Base model	5.76%	8.33%	8.88%
	Our model (α=0.3)	5.36%	8.15%	9.40%
CBOW	Base model	6.00%	8.72%	10.05%
	Our model (α=0.3)	**6.31**%	**8.92**%	**10.10**%

Table 5.2: Top-1 accuracy in the word analogies test.

new empty set and go through all vocabulary in the base model and if the word exists in our model (words with only one subword) then we add its representation to the set; otherwise, we decompose it and then add its representation according to equation 5.2. The results are summarized in Table 5.2. Our method performs as well, if not better, than the Word2Vec while being around 60% smaller in size. The difference in size come from the vocabulary size which is 155K for our model compared to 383K for the base model. We also notice that CBOW performs better than Skip-gram which is consistent with previous research findings [33].

5.5.3 OOV Handling

One important distinction that separates our model from the base model is its ability to accommodate OOV words. To test the quality of these generated OOV vectors, we go through all the OOV words in our analogy benchmark and generate vectors for the ones we can, i.e. the ones for which we have entries for all their respective subwords. The total number of OOV words is 127 and we are able to generate representations for 70 of them covering 55.12% of all the OOV words. We evaluated the performance of our best performing model (CBOW, dim=200) and found that adding the OOV representations increased the accuracy from 10.10% to 13.32%. To compare our OOV representations, we trained a fastText [17] model, a popular embedding approach that can handle OOV, and then calculated the accuracy before adding the OOV entities and after; keep in mind that the vocabulary size in both, fastText and our model, will be the same. FastText achieves 8.05% before adding the

Model	OOV handling	Top-1 accuracy
fastText	without OOV handling	8.05%
	with OOV handling	6.44%
Our model	without OOV handling	10.10%
	with OOV handling	13.32%

Table 5.3: Top-1 accuracy of our model compared to fastText when generating vectors for OOV words in the word analogies test.

	APMD	LABR	HARD
Base model	29.95%	86.18%	92.99%
Our model	**37.75%**	**86.21%**	**93.09%**

Table 5.4: Performance (accuracy) of our model compared to the base model on the three datasets.

OOV entities and 6.44% after adding them as shown in Table 5.3. Not only did fastText achieve worse gains than our model, it actually performs worse than before which calls to question the accuracy of fastText's OOV embeddings for Arabic.

5.5.4 Extrinsic Evaluation

We evaluate our approach on the three datasets mentioned above. We feed the embeddings of the words to a bidirectional Gated Recurrent Units (GRU) [25] network to train it. After that, we evaluate the network on the test set. Table 5.4 shows the performance of our model compared to the base model. We can see that our model clearly outperforms the base model in one dataset (APMD) and performs slightly better on the two other datasets (LABR and HARD).

5.6 Related Work

While many works studied the intersection of word embeddings and morphology, we do not know of any previous work that has utilized complex morphology for compression. Moreover, many of the proposed methods to incorporate morphology add complexities to the

model and are technique dependent instead of incorporating morphology by simply rethinking word sequences. Taylor and Brychcín [81] analyzed morphological relations in Arabic word embeddings. They noted that some morphological features are captured in embeddings representations. Shapiro and Duh [76] proposed utilizing subword information in training embeddings to enrich the representations and showed that it improves the performance on word similarity tasks. Salama et al. [71] investigated morphological-based embeddings and lemma-based embeddings. They utilized part-of-speech information to train their embeddings, similar to Trask et al. [83], and then build lemma-based embeddings from them by aggregating on different senses of each word first and them combining words that share the same lemma. El-Kishky et al. [31] tackled the problem of extracting roots of words and proposed an extension to fastText [17] that utilize morphemes.

5.7 Conclusion

We showed in this work that Arabic's complex morphology can be used to reduce the size of its embeddings models while still achieving good performance. Breaking words into subwords not only reduces the vocabulary size and hence the model size but also provides a simple way to produce good out-of-vocabulary representations. Our proposed approach is technique-agnostic and can be used with any word embedding technique. One possible future work that can be built upon ours is the investigation of using segmenting using word roots instead prefixes and suffixes. That would reduce the model size more but will result in more segments which will cause more complications in combining all the different subwords.

Chapter 6

Learning from Sequences in Natural Language: Rethinking Tokenization and Word Segmentation

6.1 Summary

Arabic, like other highly inflected languages, encodes a large amount of information in its morphology and word structure. In this work [5], we investigate the effect of subword segmentation strategies in BERT that take into account Arabic's relatively complex morphology. We modify the tokenization layer of Google's pretrained multilingual BERT model by incorporating information on morphology at fine-tuning time. By doing so, we achieve state of the art performance on two Arabic NLP datasets without any pretraining.

6.2 Introduction

One feature that is unique to Arabic, and other highly inflected languages, is the expressiveness of its words. The fact that Arabic encodes a large amount of information in its word

This chapter is based on work that was published at The Fifth Arabic Natural Language Processing Workshop (WANLP 2020) in The 28th International Conference on Computational Linguistics (COLING 2020).

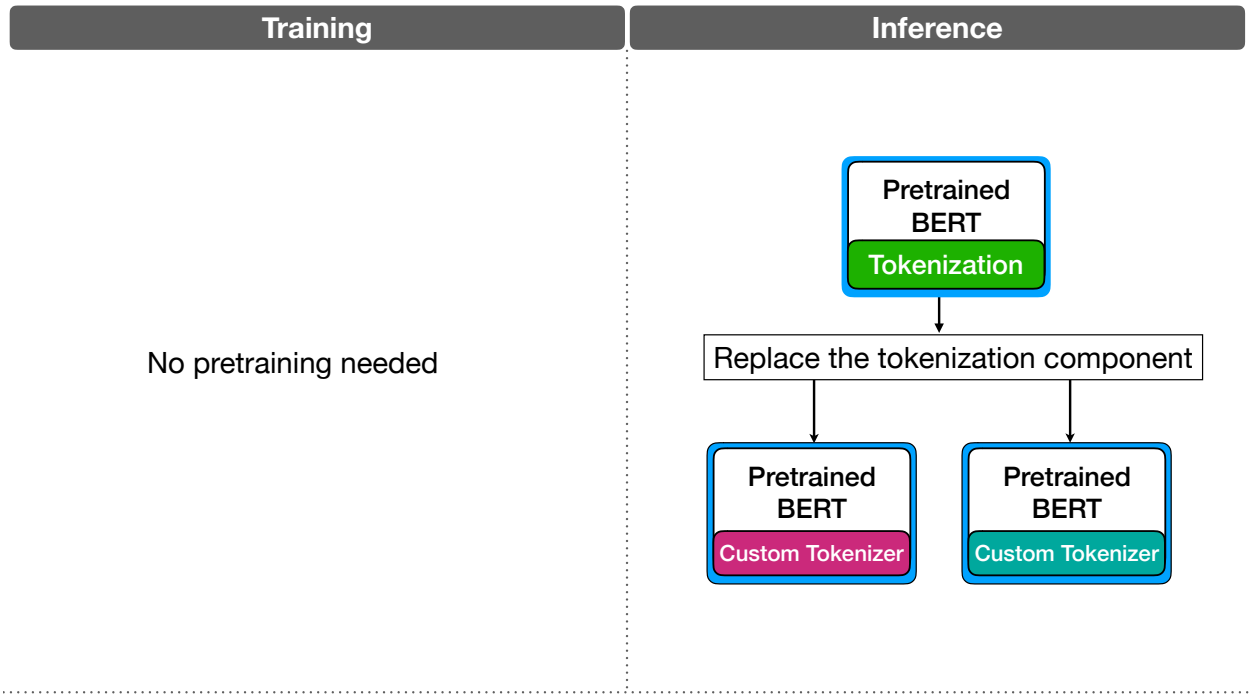

Figure 6.1: The training and inference stages of our proposed strategies.

structure leads to potential problems in learning embeddings.

In this work, we propose embedding strategies for Arabic that take into consideration its rich morphology by modifying the tokenization phase of BERT [29]. Figure 6.1 summarizes our approach and illustrates what happens in the training and inferences stages.

We investigate different tokenization schemes when using BERT without needing any pretraining, in contrast to previous approaches such as Antoun et al. [11]. We simply modify the tokenization part of Google's pretrained multilingual BERT model [29] resulting in models that:

1. Achieve state of the art results on two Arabic NLP datasets.
2. Do not require pretraining and can work on top of existing models.

The rest of the chapter is structured as follows: Section 6.3 highlights the embedding

approaches we are proposing; Section 6.4 details the experiments and the results; Section 6.5 discusses related work; and Section 6.6 concludes by summarizing our findings and pointing to potential directions for future work.

6.3 Approach

Transformer-based [85] contextual embedding models, such as BERT [29], often require a tokenization step that solves problems such as the out-of-vocabulary issue. Byte-pair encoding (BPE) [74, 36], one of the most popular tokenization methods relies on segmenting each word into the most frequent subwords. Shapiro and Duh [75] have shown that byte-pair encoding does not perform well for Arabic compared to other languages. One possible explanation of this is that byte-pair encoding does not include information derived from a given language's morphology. We did some experiments using BERT's pretrained tokenizer and found instances where the produced segments generated erroneous meanings. For example, the word *malʿab* in most cases is a noun that refers to a stadium or field. BERT tokenizes *malʿab* by segmenting it to *mal* (milliliter) and *ʿab* (gulp or fill up), instead of the correct segmentation: *ma* (a prefix used to create nouns of place) and *lʿab* (play). BERT's segmentation seems to indicate that the word *malʿab* is related to liquids and water (milliliter/gulp/fill up). Keep in mind that both *ma* and *lʿab* are in BERT's subword vocabulary. To test the quality of the two tokenizations (mal+ʿab and ma+lʿab), we manually collected words that relate to both 'stadium' and 'water': drink, fish, liter, team, sport, swimming, juice, player, goal, liquid, cup, club, liquid, and play; and got their BERT representations using both segmentations. After that we compared each word with both representations to see which one it is closer to. We notice that words related to 'stadium' are usually closer to the morphological representation as shown to Table 6.1.

We propose two methods that aim to incorporate a language's structure via better segmentations, which we call MorphBERT (Morphology BERT) and CharBERT (Character BERT). In MorphBERT, a custom tokenizer is used to replace the default tokenizer layer as seen in Figure 6.2. That custom tokenizer will use a language specific segmenter, Farasa [1]

Segmentation type	Segments	Arabic words closer to it
BERT tokenizer	mal+ʿab: (مل+عب)	drink (شرب), fish (سمكة), cup (كوب), liter (لتر), juice (عصير), liquid (سائل), and swimming (سباحة)
Morphological	ma+lʿab (م+لعب)	play (لعب), team (فريق), club (نادي), goal (هدف), player (لاعب), and sport (رياضة)

Table 6.1: BERT tokenization vs morphological segmentation for the word malʿab. (stadium)

in our case, to segment each word before processing it. We then pass each word, after segmentation, to the original model's tokenizer to make sure that the produced segments are in the model's vocabulary. Keep in mind that MorphBERT differs from AraBERTv1 [11], an Arabic BERT model that also utilizes Farasa, in that it does not require pretraining. In addition to that, Antoun et al. [11] preprocess the training corpus by segmenting it using Farasa before training AraBERTv1 which is not the case with MorphBERT.

In CharBERT, we segment everything to characters as shown in Figure 6.2. The main idea behind CharBERT is to let the network learn these language structures on its own. Both of these models do not require training and can be used with any pretrained model as we see in Figure 6.2. This is important due to the expensive — money-wise, time-wise, and environment-wise — process of training BERT and other state of the art models. We believe that developing simpler, more sustainable, and more efficient NLP models is of an utmost importance due to the many problems that arise from computationally heavy models, [80] which can take weeks to train on many TPUs/GPUs. Moreover, most people, especially in less developed countries, do not have the resources to train these models, which limits accessibility.

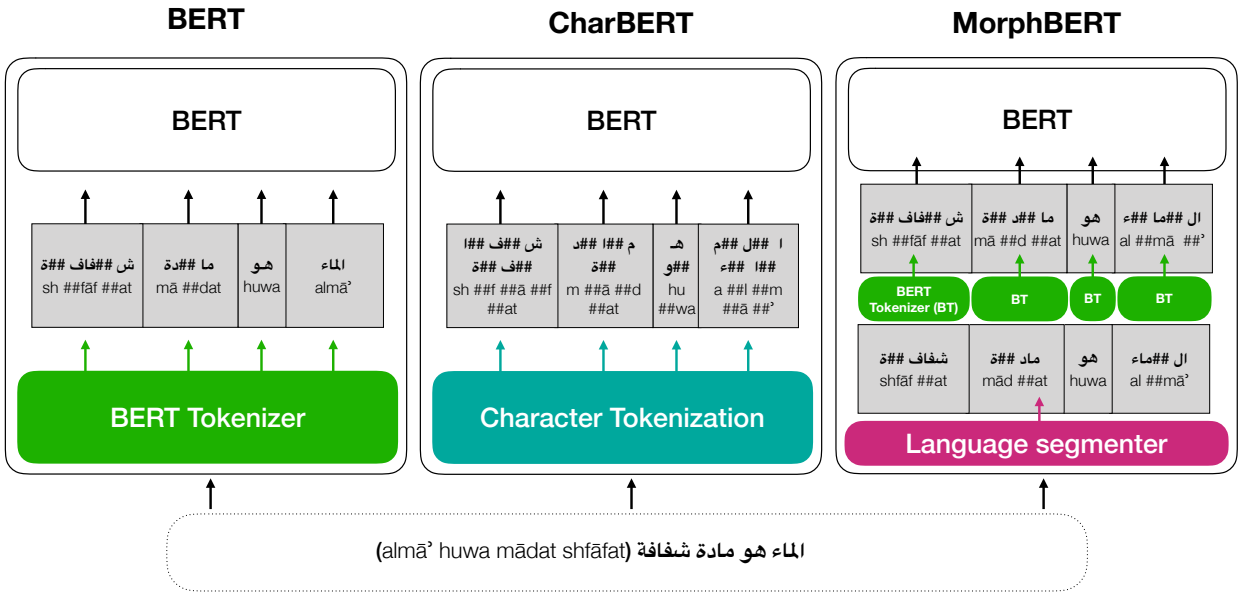

Figure 6.2: The different segmentation approaches: BERT (default tokenizer), CharBERT, and
MorphBERT.

6.4 Experiments and Results

In this section we detail our experiments and highlight the results produced by the models we proposed.

6.4.1 Evaluation Datasets

We use the following datasets to evaluate our models extrinsically:

1. **APMD**: The Arab poem meters dataset [10] which consists of 55,440 poem verses with each verse classified into one of the fourteen Arabic poetry meters. The data is split into training and testing sets.

2. **HARD**: The Hotel Arabic Reviews Dataset [32] consists of 93,700 hotel reviews that are classified into positive or negative according to their rating. Reviews with a rating of four or five were assigned positive, and those with a rating of one or two were labeled negative. Reviews with a rating of three were ignored. We split the data into 80% and 20% training and testing sets respectively using the script provided by Antoun et

al. [11].

3. **LABR**: The Large-scale Arabic Book Reviews dataset [9] consists of 63,000 book reviews rated between one and five. We use the unbalanced two-class dataset, where reviews with a rating of one or two are labeled negative, and those with a rating on four or five are labeled positive. The data is split into training and testing sets.
4. **ANERcorp**: ANERcorp [15, 11] is an Arabic named-entity recognition dataset that contains 316 annotated articles. Every token in the dataset is tagged with one of the following: location, person, organization, miscellaneous, or other.

We fine-tune the three models: the base cased multilingual BERT, MorphBERT, and CharBERT; and then compare their performances on the three datasets mentioned above. MorphBERT and CharBERT were both implemented with the Transformers library [88]. We follow the recommendations from BERT's paper [29] in setting the fine-tuning hyperparameters. We run them all for four epochs in batches of 32 or 16 depending on the lengths of input sequences to avoid memory issues on the GPU. We optimize using the Adam algorithm with a learning rate of $2e^{-5}$, $\beta 1 = 0.9$, and $\beta 2 = 0.999$. We also compare our models to AraBERT (AraBERTv0.1 and AraBERTv1) which consists of two monolingual BERT models trained on Arabic that were proposed by Antoun et al. [11] and use the results reported by them for LABR and HARD. For APMD, we downloaded their models and fine-tuned them on the task. For ANERcorp, we used the script provided by Antoun et al. [11] in fine-tuning our models. Table 6.2 shows the performance of the five models on the downstream tasks.

It is interesting that by just changing the tokenization method we can improve BERT's performance in Arabic without retraining. As we can see in Table 6.2, MorphBERT and CharBERT achieve state of the art performance on LABR and APMD respectively. The previous state of the art model for LABR is the MCE-CNN model proposed by Dahou et al. [27] which achieves an accuracy of 87.48%. Our models perform better than AraBERT in two tasks even though AraBERT was: 1) trained specifically for Arabic, and 2) trained on a larger Arabic corpus: 24GB of data for AraBERT compared to 4.3GB for the multilingual

	APMD	LABR	HARD	ANERcorp
metric	accuracy	accuracy	accuracy	macro-F1
mBERT	70.65%	85.85%	95.96%	78.4%
MorphBERT	71.03%	**89.87%**	95.86%	79.86%
CharBERT	**84.09%**	85.89%	95.67%	71.76%
AraBERTv0.1	70.02%	85.90%	96.20%	**89.17%**
AraBERTv1	60.99%	86.70%	**96.20%**	88.67%

Table 6.2: Performance of MorphBERT and CharBERT compared to the multilingual BERT (mBERT), AraBERTv0.1, and AraBERTv1

BERT. Although mBERT was trained on over a hundred languages, simply replacing tokenizations allowed us to add language specific information without requiring any training. While normally byte pair encoding learns representations of subwords without paying attention to their meaning, we can utilize this procedure of breaking words into chunks that make more sense as we saw in the *mal'ab* example before. CharBERT in particular is interesting; one would expect that it will require more time to fine-tune since it only uses characters. Nevertheless, it achieves great performance without requiring more epochs than the other methods. One potential issue with CharBERT is that it results in very long sequences due to the character segmentation approach it follows which lead to more frequent truncations than other models. One potential way to mitigate this is by using new models such as Longformer [14] that allow longer sequences than BERT.

Previous research [86, 11, 87, 57] has shown that a language specific BERT model performs better than a multilingual one. This is the first work, according to our knowledge, that shows that by tweaking a multilingual BERT model one can beat a BERT model trained on a specific language. Natural language processing entered a new era with the advent of pretrained models that do not need to be trained from scratch for every task but can simply be tweaked/fine-tuned instead. Our results show that it may be possible to only have one multilingual model that can be tweaked instead of learning a pretrained model for every

language.

6.5 Related Work

Many works have noted how sensitivity to Arabic's morphological complexity can result in better performance in standard NLP tasks. However, we do not know of any previous work that has specifically focused on the effect of different subword segmentation strategies on different Arabic BERT embedding models. Antoun et al. [11] trained an Arabic specific BERT model. They also trained another Arabic BERT model in which they segment the text before training the model and showed that it usually improves performance. Shapiro and Duh [75] showed that BPE performs worse for Arabic compared to other languages.

6.6 Conclusion

We show that tokenization that pays attention to Arabic's morphology can create better contextual embedding models. We also show the importance of tokenization in BERT where we were able to achieve impressive performance without requiring any pretraining. One possible future work would be to investigate tokenization's effect in other morphologically rich languages such as Hebrew and Turkish and see if our results can be generalized to other highly inflected languages.

Chapter 7

Conclusion and Future Directions

The goal of this dissertation was to show that learning from sequences can be a powerful tool beyond classical word embeddings. Our work has shown that we can learn good representations from sequences in more complicated environments, whether we are tackling educational representations, identifying narrators in historical contexts, or rethinking representations of morphologically complex languages. Our work opens the doors to research employing sequence-driven representations and abstractions to a wide array of different fields and domains that have their own challenging complex representations.

We will be concluding this dissertation by highlighting some interesting future directions of the work presented:

7.0.1 Learning from Sequences in Education: Representing Academic Degrees

The educational embeddings we proposed achieve good results at solving a daunting task in education today as we show in Chapter 3. There are many possible future works that can build upon our research. One is to use our representations to build personalized learning assistants and help in automating academic advising. Another possible future work is investigating degree representations in multiple universities and checking whether we can learn a translation/mapping between universities similar to how machine translation is done between natural languages.

7.0.2 Learning from Sequences in History: Identifying Narrators in Classical Arabic Texts

The automatic narrator identifier we built from sequences of narrators is the first of its kind. Moreover, we illustrated its capabilities by correcting an error in one of the largest historical Arabic texts corpora as we showed in Chapter 4. One way our work can be extended is to investigate entity linking via sequences more and observe how they perform on different datasets. Another potential work is to train a classical Arabic BERT model instead of using a generic pretrained model and see how that affects performance. In addition to that, looking into approaches that incorporate some of the knowledge base biographical details into the model and see if that improves the performance.

7.0.3 Learning from Sequences in Natural Language: Incorporating Morphology in Traditional Word Embeddings

The embeddings we proposed in Chapter 5 excelled in different aspects (performance, size, and out-of-vocabulary handling) compared to current embedding techniques. One possible future work that can be built upon ours is the exploration of segmenting using word roots instead prefixes and suffixes. That would likely lead to an extra reduction in the model size. Other possible future work concerns more sophisticated exploration of combining morphological features and word embeddings

7.0.4 Learning from Sequences in Natural Language: Rethinking Tokenization and Word Segmentation

In Chapter 6, we showed how we can get better representations by simply tweaking how word segmentation is done. This discovery led us to two state-of-the-art models. One possible future work would be to investigate the effect of tweaking word segmentations in other morphologically rich languages such as Hebrew and Turkish and see if our results

can be generalized to other highly inflected languages. Another interesting future work is to explore word segmentation more and come up with strategies to produce better word segments without needing to use an external language-specific tool for each language.

References

[1] Abdelali, A., Darwish, K., Durrani, N., and Mubarak, H. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations* (San Diego, California, June 2016), Association for Computational Linguistics, pp. 11–16.

[2] Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., and Zou, J. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569* (2019).

[3] Al-Ayyoub, M., Nuseir, A., Alsmearat, K., Jararweh, Y., and Gupta, B. Deep learning for arabic nlp: A survey. *Journal of computational science 26* (2018), 522–531.

[4] Alkaoud, M., and Pardos, Z. A. Degree curriculum contraction: A vector space approach. In *Artificial Intelligence in Education* (Cham, 2019), S. Isotani, E. Millán, A. Ogan, P. Hastings, B. McLaren, and R. Luckin, Eds., Springer International Publishing, pp. 14–18.

[5] Alkaoud, M., and Syed, M. On the importance of tokenization in Arabic embedding models. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop* (Barcelona, Spain (Online), Dec. 2020), Association for Computational Linguistics, pp. 119–129.

[6] Allamanis, M., Barr, E. T., Devanbu, P., and Sutton, C. A survey of machine learning for big code and naturalness. *ACM Comput. Surv. 51*, 4 (July 2018).

[7] Alon, U., Zilberstein, M., Levy, O., and Yahav, E. code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages 3*, POPL (2019), 1–29.

[8] Altammami, S., Atwell, E., and Alsalka, A. Text segmentation using n-grams to annotate hadith corpus. In *Proceedings of the 3rd Workshop on Arabic Corpus Linguistics* (Cardiff, United Kingdom, July 2019), Association for Computational Linguistics, pp. 31–39.

[9] Aly, M., and Atiya, A. LABR: A large scale Arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Sofia, Bulgaria, Aug. 2013), Association for Computational Linguistics, pp. 494–498.

[10] Alyafeai, Z. ARBML. `https://github.com/zaidalyafeai/ARBML`, Jan. 2020.

[11] Antoun, W., Baly, F., and Hajj, H. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection* (Marseille, France, May 2020), European Language Resource Association, pp. 9–15.

[12] Azmi, A. M., Al-Qabbany, A. O., and Hussain, A. Computational and natural language processing based studies of hadith literature: a survey. *Artificial Intelligence Review 52*, 2 (2019), 1369–1414.

[13] Azmi, A. M., and Badia, N. B. itree - automating the construction of the narration tree of hadiths (prophetic traditions). *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering(NLPKE-2010)* (2010), 1–7.

[14] Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv:2004.05150* (2020).

[15] Benajiba, Y., and Rosso, P. Anersys 2.0: Conquering the ner task for the arabic language by combining the maximum entropy with pos-tag information. In *IICAI* (2007), pp. 1814–1823.

[16] Bird, S., Klein, E., and Loper, E. *Natural language processing with Python*, 1st ed ed. O'Reilly, Beijing ; Cambridge [Mass.], 2009. OCLC: ocn301885973.

[17] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* (2016).

[18] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146.

[19] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems* (2016), pp. 4349–4357.

[20] Bounhas, I. On the usage of a classical arabic corpus as a language resource: Related research and key challenges. *ACM Trans. Asian Low-Resour. Lang. Inf. Process. 18*, 3 (Jan. 2019).

[21] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. Extensible Markup Language (XML) 1.0 (Fifth Edition). `https://www.w3.org/TR/REC-xml/`, Nov. 2008. Accessed: 2020-03-05.

[22] Brown, J. A. *Hadith: Muhammad's legacy in the medieval and modern world*. Oneworld Publications, 2017.

[23] Caliskan, A., Bryson, J. J., and Narayanan, A. Semantics derived automatically from language corpora contain human-like biases. *Science 356*, 6334 (2017), 183–186.

[24] Chiu, J. P., and Nichols, E. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics 4* (2016), 357–370.

[25] Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 103–111.

[26] CIA. Legal system. `https://www.cia.gov/the-world-factbook/field/legal-system/`, Mar. 2021. Accessed: 2021-03-03.

[27] Dahou, A., Xiong, S., Zhou, J., and Elaziz, M. A. Multi-channel embedding convolutional neural network model for arabic sentiment classification. *ACM Trans. Asian Low-Resour. Lang. Inf. Process. 18*, 4 (May 2019).

[28] de Wynter, A., and Perry, D. J. Optimal subarchitecture extraction for bert. *CoRR abs/2010.10499* (2020).

[29] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[30] Efstathiou, V., Chatzilenas, C., and Spinellis, D. Word embeddings for the software engineering domain. In *Proceedings of the 15th International Conference on Mining Software Repositories* (2018), pp. 38–41.

[31] El-Kishky, A., Fu, X., Addawood, A., Sobh, N., Voss, C., and Han, J. Constrained sequence-to-sequence Semitic root extraction for enriching word embeddings. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop* (Florence, Italy, Aug. 2019), Association for Computational Linguistics, pp. 88–96.

[32] Elnagar, A., Khalifa, Y. S., and Einea, A. *Hotel Arabic-Reviews Dataset Construction for Sentiment Analysis Applications*. Springer International Publishing, Cham, 2018, pp. 35–52.

[33] Elrazzaz, M., Elbassuoni, S., Shaban, K., and Helwe, C. Methodical evaluation of Arabic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Vancouver, Canada, July 2017), Association for Computational Linguistics, pp. 454–458.

[34] Farghaly, A., and Shaalan, K. Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP) 8*, 4 (2009), 1–22.

[35] Firth, J. R. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis* (1957).

[36] Gage, P. A new algorithm for data compression. *C Users Journal 12*, 2 (1994), 23–38.

[37] Glicksberg, B. S., Miotto, R., Johnson, K. W., Shameer, K., Li, L., Chen, R., and Dudley, J. T. Automated disease cohort selection using word embeddings from electronic health records. In *PSB* (2018), World Scientific, pp. 145–156.

[38] Gordon, A. 'MicroMasters' surge as MOOCs go from education to qualification. `forbes.com/sites/adamgordon/2018/02/13/voice-of-employers-rings-out-as-moocs-go-from-education-to-qualification`, Feb. 2018. Accessed: 2019-01-28.

[39] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (2018).

[40] Gude, A. Python2Vec: Word embeddings for source code. `gab41.lab41.org/python2vec-word-embeddings-for-source-code-3d14d030fe8f`, Mar. 2018. Accessed: 2020-04-16.

[41] Harris, Z. S. Distributional structure. *Word 10*, 2-3 (1954), 146–162.

[42] Hindle, A., Barr, E. T., Su, Z., Gabel, M., and Devanbu, P. On the naturalness of software. In *2012 34th International Conference on Software Engineering (ICSE)* (2012), IEEE, pp. 837–847.

[43] Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[44] Islamweb. Gawāmiʿal-kalim. `https://gk.islamweb.net`, Mar. 2021. Accessed: 2021-03-03.

[45] ITU. Country ICT Data. `https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx`, Dec. 2019. Accessed: 2020-03-04.

[46] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651* (2016).

[47] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[48] Khattak, F. K., Jeblee, S., Pou-Prom, C., Abdalla, M., Meaney, C., and Rudzicz, F. A survey of word embeddings for clinical text. *Journal of Biomedical Informatics: X* (2019), 100057.

[49] Kozlowski, A. C., Taddy, M., and Evans, J. A. The geometry of culture: Analyzing the meanings of class through word embeddings. *American Sociological Review 84*, 5 (2019), 905–949.

[50] KS, K., and Sangeetha, S. Secnlp: A survey of embeddings in clinical natural language processing. *arXiv preprint arXiv:1903.01039* (2019).

[51] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (San Francisco, CA, USA, 2001), ICML '01, Morgan Kaufmann Publishers Inc., p. 282–289.

[52] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[53] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[54] Ma, X., and Hovy, E. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 1064–1074.

[55] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[56] Maraoui, H., Haddar, K., and Romary, L. Segmentation tool for hadith corpus to generate tei encoding. In *International Conference on Advanced Intelligent Systems and Informatics* (2018), Springer, pp. 252–260.

[57] Martin, L., Muller, B., Ortiz Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D., and Sagot, B. Camembert: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020).

[58] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[59] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[60] Motzki, H. Dating muslim traditions: A survey. *Arabica 52*, Fasc. 2 (2005), 204–253.

[61] Muazzam, A., Siddiqui, M., Saleh, M., Saleh, A., and Bagais. Extraction and visualization of the chain of narrators from hadiths using named entity recognition and classification. *International Journal of Computational Linguistics Research 5* (04 2014).

[62] Muther, R., and Smith, D. Tracing traditions: Automatic extraction of isnads from classical Arabic texts. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop* (Barcelona, Spain (Online), Dec. 2020), Association for Computational Linguistics, pp. 130–138.

[63] Nguyen, D. Q., Vu, T., and Tuan Nguyen, A. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (Online, Oct. 2020), Association for Computational Linguistics, pp. 9–14.

[64] Office of Planning & Analysis, University of California, Berkeley. *Majors and Minors of Degree Recipients, 2010-11 to 2014-15*, 2016.

[65] Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

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

[67] Rabiner, L., and Juang, B. An introduction to hidden markov models. *IEEE ASSP Magazine 3*, 1 (1986), 4–16.

[68] Ramshaw, L. A., and Marcus, M. P. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*. Springer, 1999, pp. 157–176.

[69] Řehůřek, R., and Sojka, P. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (Valletta, Malta, May 2010), ELRA, pp. 45–50. `http://is.muni.cz/publication/884893/en`.

[70] Ruder, S. Why You Should Do NLP Beyond English. `http://ruder.io/nlp-beyond-english`, 2020.

[71] Salama, R. A., Youssef, A., and Fahmy, A. Morphological word embedding for arabic. *Procedia computer science 142* (2018), 83–93.

[72] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[73] Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. Green ai. *Commun. ACM 63*, 12 (Nov. 2020), 54–63.

[74] Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* (2015).

[75] Shapiro, P., and Duh, K. Bpe and charcnns for translation of morphology: A cross-lingual comparison and analysis. *arXiv preprint arXiv:1809.01301* (2018).

[76] Shapiro, P., and Duh, K. Morphological word embeddings for Arabic neural machine translation in low-resource settings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models* (New Orleans, June 2018), Association for Computational Linguistics, pp. 1–11.

[77] Sharir, O., Peleg, B., and Shoham, Y. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900* (2020).

[78] Smith, N. A. Contextual word representations: A contextual introduction. *arXiv preprint arXiv:1902.06006* (2019).

[79] Soliman, A. B., Eissa, K., and El-Beltagy, S. R. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science 117* (2017), 256–265. Arabic Computational Linguistics.

[80] Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 3645–3650.

[81] Taylor, S., and Brychcín, T. The representation of some phrases in arabic word semantic vector spaces. *Open Computer Science 8*, 1 (2018), 182–193.

[82] Theeten, B., Vandeputte, F., and Van Cutsem, T. Import2vec: Learning embeddings for software libraries. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)* (2019), IEEE, pp. 18–28.

[83] Trask, A., Michalak, P., and Liu, J. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388* (2015).

[84] Turc, I., Chang, M.-W., Lee, K., and Toutanova, K. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2* (2019).

[85] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, ., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.

[86] Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J., Salakoski, T., Ginter, F., and Pyysalo, S. Multilingual is not enough: Bert for finnish, 2019.

[87] Vries, W. d., Cranenburgh, A. v., Bisazza, A., Caselli, T., Noord, G. v., and Nissim, M. BERTje: A Dutch BERT Model. *arXiv:1912.09582 [cs]* (Dec. 2019).

[88] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (Online, Oct. 2020), Association for Computational Linguistics, pp. 38–45.

[89] Yang, J., and Zhang, Y. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (2018).

[90] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems 33* (2020).

[91] Zhang, Y., Rahman, M. M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., McNamara, Q., Angert, A., Banner, E., Khetan, V., et al. Neural information retrieval: A literature review. *arXiv preprint arXiv:1611.06792* (2016).