Utrecht University

Evaluation of state-of-the-art machine learning approaches on the detection of variations for entity mentions

Master thesis

Artificial Intelligence

Department of Information and Computing Sciences

Internal first supervisor
Dr. Marijn Schraagen

External supervisor
Dr. Jelte Mense

Author
Mustafa Al Bawi

Internal second supervisor
Prof. dr. C.J. (Kees) van Deemter

December 11, 2020

Abstract

Natural Language Processing is concerned with the interactions between computers and natural languages. Named Entity Recognition is an important subject within the field of Natural Language Processing. Finding entities within texts is the main goal of Named Entity Recognition, this can be done in numerous ways. Another important task in Natural Language Processing is Entity Linkage, in which unique entities are assigned to entities in texts. Although research on Natural Language Processing has come a long way, we still experience issues recognizing and linking entities in modern-day texts. Modern-day texts, used within social media, often consist of short texts which provide limited context and make it harder to find and link entities. Not only do modern-day texts lack enough context, but they are also written by various authors. Having text which is written by various authors gives rise to the entity name variation problem. Having many possible mention variations, with little contextual information, is why short texts are often seen as troublesome for many recognition and linking tasks. In this research, we will look at name variations for names in short texts. Semitic names often have a huge amount of variations when transliterated to Latin script, that is why we will try to look at some Semitic names written in English to test our hypotheses. To link entity mention variations we need to be able to classify a tweet based on the entity that is mentioned. One way to classify tweets based on mentions of an entity, regardless of variation, is to look at the context of an entity. In our research, we will try and research three methods that can be used to correctly link entities in the presence of name variations found in short texts. Firstly we will try and see how well we can link entities in the presence of name variation using a Logistic Regression classifier. Then we will try to link entity name variations by classification using a Convolutional Neural Network. Finally, we will look at Topic modeling as an approach to cluster our short texts allowing us to group variations of entity names. We test our models on various tasks in which we test the influence of the number of entities, the number of variations, and the available context. For our research, we will be using different types of datasets. Our first dataset contains 10 unique entities with names from Semitic origin, found in 3874 unique tweets. A second dataset is also used which contained 753 tweets with 2 entities that have the same entity mention, accompanied by a group of unknown entities with the same entity mentions. Our research shows that depending on the number of entities and training knowledge, we can choose a model that might fit best for each situation. With a higher amount of entities, it seems that Convolutional Neural Network models perform better than the other models. In a low number of entities, we can show that Topic Modeling might become a better alternative to classify entities. We also look at how our models behave when trying to differentiate between entities with the same names. For example, tweets containing the same entity mention "Muhammad" whilst referring to different entities. Other research on the topic of entity linking is often based on different types of data or different datasets, this means that other research on entity linking is not suitable to compare with our data. To get an indication of how well our models work, we will compare our models with baseline models that we created. Our research showed us that Topic Modeling had promising results when differentiating entities with the same name.

Contents

Page

1 Introduction . . . 3
2 Background . . . 6
3 Methodology . . . 10
3.1 Dataset . . . 11
3.1.1 Dataset splits . . . 13
3.1.2 Conditions . . . 14
3.1.3 Preprocessing . . . 15
3.2 Baseline Models . . . 15
3.3 Logistic Regression . . . 17
3.4 Short Text Topic Modeling . . . 19
3.5 Convolutional Neural Network . . . 20
4 Results . . . 22
4.1 Results for Baseline models . . . 22
4.2 Results for 10 entities . . . 26
4.2.1 Linear Regression Model . . . 26
4.2.2 Convolutional Neural Network Model . . . 27
4.2.3 Gibbs Sampling Dirichlet Multinomial Mixture Model . . . 30
4.3 Results for 5 entities . . . 32
4.3.1 Linear Regression Model . . . 32
4.3.2 Convolutional Neural Network Model . . . 33
4.3.3 Gibbs Sampling Dirichlet Multinomial Mixture Model . . . 36
4.4 Results for Muhammad dataset . . . 38
4.4.1 Linear Regression Model . . . 38
4.4.2 Convolutional Neural Network Model . . . 39
4.4.3 Gibbs Sampling Dirichlet Multinomial Mixture Model . . . 41
4.5 Overview of model results . . . 43
5 Conclusion and Discussion . . . 49
5.1 Model performance . . . 49
5.2 Entity mention variations . . . 50
5.3 Limitations and Future Work . . . 51
5.3.1 Generalizability . . . 51
5.3.2 Dataset . . . 52
5.3.3 Language . . . 52
5.3.4 Models and features . . . 52
5.3.5 Real world data . . . 53
5.3.6 Possible improvements . . . 54
5.4 Practical relevance . . . 54
A Appendix . . . 56
A.1 Entity mentions . . . 56
A.2 10 entities . . . 57
A.2.1 LR model . . . 57
A.2.2 CNN model . . . 58
A.3 5 entities . . . 60
A.3.1 LR model . . . 60
A.3.2 CNN model . . . 61
A.4 Muhammad Dataset . . . 63
A.4.1 LR model . . . 63
A.4.2 CNN model . . . 64

1 Introduction

Within our modern age, communication methods have changed. Messages can be sent directly using different tools and platforms. Users often tend to write fast, make mistakes, and communicate their message knowing their recipient understands the underlying implications. In some cases, users of instant messaging boards or services might use code names or words to avoid being detected. Organized crime has been a thorn in societies for a very long time and the disruption caused by criminal activities from organized groups can be catastrophic. With the use of our new form of communication, organizations can send messages without ever having physical contact. New ways of communicating imply that new ways of detecting people are needed. With digital communication being more and more present in our current society, designing tools that can help identify information or name variants of entities could be very useful for law enforcement, governments, and even companies. The speed at which such a tool operates is also of great importance. A tool that deciphers whether something or someone has any associations with a terrorist group or other organized crime needs to be able to classify quickly. Creating a system that can help find the correct entity in a fast and efficient manner, can be done using techniques from the field of Artificial Intelligence (AI). Users are found all around the world and, therefore, communicate using many languages. Texts used to gain information on an entity might have been transliterated or written down by non-native speakers. The transliteration and translation of foreign texts often come with a loss of information. Transliteration in particular gives rise to name-ambiguity. Due to a lack of consensus, when a name is transliterated from a different script it is often attributed with different variations. For example, when we look at Arabic names transliterated to English or other languages using the Latin script, we can see different transliterations for the same name: Muhammed, Muhamed, Muhamad. There are various reasons why the different name transliterations occur. Most of the Latin variations found of Arabic names are likely due to the variation in the pronunciation of the name in different Arabic dialects. The lack of specified vowels and consonants is also a contributing factor for the different transliterations of Arabic names. Another issue that might arise when trying to extract information from texts is the lack of context. Not only is detection an issue, but knowing whether the same entities are mentioned in two different pieces of small texts can also be a challenging task. Being able to find variations of an entity mention is only one of the multiple possible uses for name variation detection. Better classification of entities in short texts could potentially improve already existing systems. Previous research on the field of entity linking and classification in written texts is often done using existing databases or information. However, there is little to no research found on linking entities without any existing databases or information, which prompted us to research our subject.

For this project, we will look at how well we can classify short texts containing entity mentions. In this thesis, we will explore how different models can classify selected entities within a short text. To be more precise, we want to look at how Logistic Regression (LR) models, Topic Modeling (TM) models, and Convolutional Neural Networks (CNN) can assist in classifying tweets that contain entity mentions. We research our main goal by trying to answer four underlying research questions. We will be using tweets for our data, which we gathered using the Twitter API. More information on our datasets will follow in section 3.1. We will first give a short overview of our research goals followed by a more detailed description of each of these research goals. The goals of our research are :

Main research goal:

Find out which model performs best for tweet entity classification between Logistic Regression, Convolutional Neural Networks, and Topic Modeling.

Sub-questions:

1. What influence does the number of entities, on which the models are trained, have on the performance of models?
2. How well can our models detect entity variations within tweets?
3. What influence does the surrounding text of an entity have on the performance of our models?
4. Can our models correctly classify entities with similar entity mentions within tweets?

Sub-questions

1. With our first research question, we want to find out what influence the number of entities has on the performance of models. To research the influence of the number of entities, we will test our models on multiple datasets and multiple entities. We will test the models on how well they can classify 10 entities and how well they can classify only 5 entities. The entities are fairly well distinguishable. By altering the number of entities we can observe the difference in the performance of models when the amounts of entities are halved. We will then use another dataset to classify up to 2 entities. We will test our models on whether they can identify one or both of the entities. We will also test whether the models wrongfully classify any other unknown entities as being one of the two entities. The last dataset on which we will test our models contains unique personas with similar name mentions. We will perform similar tasks as on the previous datasets. The last dataset will contain unique personas, but similar name mentions. We will not try to compare our last dataset with the first 2 mentioned as we are using different entities and these are not comparable. The aforementioned datasets will be described in more detail in section 3.1. Using as large a dataset as possible would be the most optimal for our research, however, we were not able to obtain more entities due to constraints and limits of our entity gathering tools. Other issues that we have encountered, such as annotating of data and finding suitable names, have also limited our entity volume. Nonetheless, we do believe that using an exploratory dataset with the number of entities we have used will suffice for answering our research question. Multi-classification tasks using clustering tools can be very time intensive which can already be detected using our datasets.

2. For our second research question, we will split our data into training and testing sets in multiple ways. By performing different split types on our first dataset we will be able to research how well our models can detect entity variations on which they have not been trained. We test our splitting methods on both the 10 entity dataset and the 5 entity dataset, allowing

us to see if the number of entities severely impacts our model performance when detecting new entities. Our second research question allows us to see how well our models might detect new entity mentions, allowing us to perhaps find new information on an entity. Looking at ways to detect new variations of an entity is a very useful tool, bringing new insights into the field of entity variation detection.

3. With our third research question, we want to find out how the entity context might affect the model's performance. The context of our entities here refers to the surrounding words found in the short text. To research the influence of context, we will apply different conditions to our datasets. The conditions we apply to our datasets will affect our datasets by removing the entity mentions completely or replace all unique entity mentions with one single mention for all. Removing the entities completely will leave us with only the surrounding context, giving us an indication of how influential the use of context is for our models. We will be applying our conditions on all of our datasets, giving us an indication of how context could also impact our previous research questions.

4. For our fourth and last research question will determine whether our models can distinguish between entities that have the same mentions in a text. As we mentioned before, we will be using two different datasets for our research. One dataset will be containing many different name mentions for each entity, the other dataset will contain similar name mentions for 2 entities. The second dataset will allow us to research our final research question more thoroughly. Our final research question is connected to our previous question. We will investigate the performance of the models within this specific case, which might bring valuable insights into the influence of context within identity recognition.

In this thesis, we will start by describing the background information relevant to our research. In our Methods section we will explain the models and methods we have used to reach our goals, we will explain the conditions that were used and the type of data that was used. Afterward, we will report all our main results for each model, we finish our results section by giving a clear overview of our model's performances. Finally, we discuss our findings in our discussion and conclusion section. The Git repository for the project can be found on https://git.science.uu.nl/m.albawi/thesis-2020.

2 Background

The area of Natural Language Processing (NLP) is concerned with the interactions between computers and natural languages. Previous research on NLP has shown promising results when trying to extract information from linguistic data such as medical files. Finding named-entities is a very popular theme within the field of NLP and has shown good progress thus far. Since the start of this century, we have seen very efficient and effective models of unsupervised learning that can accurately classify entities in text (Cucchiarelli & Velardi, 2001). Most state-of-the-art named-entity recognizers (NER) rely a lot on hand-crafted features. Because of their dependency on hand implemented features, a lot of these NER's are often only used for a single language. Sometimes the features are even more restricted and only work for specific data types, such as books or journals only (Nadeau & Sekine, 2007). The restricted applicability and difficulty to use in general cases has prompted researchers to find more universally applicable NER's to use for multiple languages and datatypes. Besides Named-Entity Recognition, the fields of Named-Entity linking (NEL) and Record Linkage (RL) are equally important in the field of NLP. For Entity Linking the main goal is to try and discover mentions of an entity within a text and link them to a person in a knowledge base. Numerous systems have been designed to try and connect mentions of entities to unique personas in knowledge bases such as Wikipedia. Research on NEL systems has shown that there are many ways to improve existing systems by using different search strategies or including co-reference into the systems. (Hachey et al., 2013).

Record linkage tasks try to find records in a database or dataset that refer to the same unique entity in different data sources. In early research, record linkage was often used in health sectors to link patient medical records. Linkage of patient records could be done by comparing quasi-identifying information such as name, date of birth, and other information found on the records (Christen, 2012, Chapter 1.1). Record linkage techniques can even be used to link family members and create family reconstructions from databases containing personal information, like hospital files or historical archives (Schraagen & Kosters, 2014). Often, the names in databases in record linkage tasks are found in a structured form. The data is found in tables and columns, allowing researchers to easily gather information from the databases. For record linkage, named entity recognition is not needed because names can be accessed easily in the structured databases. Comparisons in record linkage tasks are, therefore, often done by measuring the difference found between selected strings of information in specified records. The most popular string distance metric, that is used to measure the difference between strings, is the Levenshtein Distance. The Levenshtein distance is the minimum number of single-character edits needed to change a word into another (Levenshtein, 1966). Levenshtein distance can be a very versatile tool used in many different fields of research. In linguistics, we see the Levenshtein distance often used to measure how related different dialects and languages are. Researchers compare standard word lists to measure linguistic distance, which is used to establish phylogenetic relationships among languages (Wichmann et al., 2010). Chowdhury et al. (2013) have also shown that online handwriting can be recognized with the help of Levenshtein distance. Shape and position information are both used by encoding them and assigning weights based on the information, a Levenshtein distance metric is then used to compute the similarity between unknown character samples and each training sample based on shape and position. It is also possible to find uses for Levenshtein distance in fields such as biology, where Chakrabarti et al. (2013) have shown that DNA sequence alignment techniques can be developed with the use of Levenshtein distance in combination with a Hidden Markov Model. Other examples of string distance measures that have been designed and used by other researchers are the Damerau-Levenshtein distance, Longest Common Subsequence (LCS) distance, the Hamming distance, and Jaro-Winkler (JW) distance. The different string measures are similar but have slight alterations. The Levenshtein distance allows deletion, insertion, and substitution. The Damerau-Levenshtein distance is similar to the Levenshtein distance but allows the transposition of two adjacent characters. The LCS distance allows only insertion and deletion. The Jaro-Winkler distance seems very different at first look because it seems to allows transposition only. However, just because the JW distance doesn't specifically look for the other edits such as deletion and insertion of characters, doesn't mean that they do not influence the algorithm. The Jaro-Winkler string comparison method tries to look more at similarity instead of the difference between strings. The JW algorithm exists of two parts: one part which counts the number of similar common characters in half the length of the longest string, and a part that increases similarity if the beginning of strings is similar (Christen, 2012, Chapter 5.5). From all the possible choices we do believe that Levenshtein is best suited, given its success in many NLP tasks. Levenshtein matching should be a good metric to compare how well most of our models perform as it is often used to match names and spelling variants (Lhoussain et al., 2015).

Most techniques that are trying to retrieve information from texts, such as NER, NEL, and RL are dependent on the surrounding context. In many cases, the surrounding context is sufficient as often large pieces of text are used. Databases often contain a vast amount of information on specific entities and can help algorithms find or link entities. However, some datasets do not have the information neatly organized and do need techniques to help find entities. The string measuring methods explained in the last paragraph could be used for NEL, but using string measuring tools for these tasks can only be done in combination with a named entity recognizer. If we take an example sentence, in which we were looking for the entity Yitzhak Rabin, such as :

> *"The Mossad in Israel quickly obtained information about the hostages, the situation on the ground, and where they were being held. They relayed this information to Israel's Prime Minister Isaac Rabin"*

We can see that there is a Levenshtein distance of 6 when comparing our sought entity and the entity in the text. However, if we don't have a tool to guide us where the entity is, we would have to compare all the words and word combinations to find a match. Not only would comparing all words and combination in the message be very time consuming, but it could also yield results in which non-relevant words or names are found with a similar Levenshtein distance. For example, if the entity Yitzhak Shamir is compared to the entity Yitzhak Rabin, we find a Levenshtein distance of 4 which is even less than the one found in the example sentence. A Levenshtein distance-based model would find a match and declare the entity mentions belonging to one entity even though they both refer to different entities. Nonetheless, some techniques can be used on full texts without having to find the entities in the texts. Word embedding tools are an example of tools that are often used to help models retrieve context information from texts. Word embedding techniques allow models to use words or tokens surrounding an entity or subject to be used as information to help classifying certain entities or subjects. The surrounding words and tokens can be used directly or changed into more practical data such as vectors. The word embedding techniques are a collection of feature learning and language modeling techniques. With the use of Word embedding techniques, researchers have been able to improve the ability of networks to learn from data. Ganguly et al. (2015) shows that it is possible to improve retrieval effectiveness using a very popular word embedding technique, Word2Vec. Word2Vec is a technique used to transform words into vectors. Transforming words into vectors can be a very helpful tool for many machine learning models, as the data is being represented as a lower-dimensional vector. We will explain Word2vec in more detail in section 3.3 where we use it as a tool to improve our model. There are also other word embeddings used that can show improvement in NLP tasks. Syntactic parsing of grammar categories has also been improved using Compositional vector grammars (Socher et al., 2013). It is also possible to use vector representations to achieve interesting results in other fields of research. Asgari and Mofrad (2015) have shown that feature extraction for biological sequences can be represented using an embedding tool named BioVec. Representing biological sequences with vectors allows deep learning models and other machine learning models to use the biological data, giving researchers the ability to apply these methods towards solving their research questions and goals. With the use of embedding tools, we can give each word in our text vectors and use these as features for our models. Having vectors for all words, thus the names in the texts, allows us to compare similarity between words, and we won't need to specify names using named entity recognition tools.

Another interesting topic that we have not yet discussed is that of text length. Many existing state-of-the-art models show a decline in performance when looking at modern datasets and texts. Modern-day communication often happens with shorter text, we share our thoughts, feelings, and ideas with multiple people from around the world in seconds. Nowadays most people do not write long letters as much as they used to, they write frequently and in a shorter manner. Yet, we are almost always able to identify any entities within our tiny texts without any issues. NER tools can perform well on normal texts in English, as grammatical rules allow only certain places for entities to exist. Patterns are often used and help NER tools to find entities in normal texts. However, texts found that are written in non-formal settings, such as social media, can be troublesome for NER tools. When identifying mentions of entities within short and unstructured texts, most State-of-the-art NER tools' performance quickly declines. The difficulties that arise, when looking at shorter texts, have sparked new interest in the field of NLP. Researchers have started to perform entity recognition on databases such as Twitter. Performing tasks on short texts have shown to be tougher than on longer texts, context can be very helpful for deciphering entities because of the added background information about a named entity. Analysis of state-of-the-art NER and NEL approaches show that the techniques do not perform robustly on micro-blog texts such as Twitter (Derczynski et al., 2015). Nonetheless, some researchers have shown moderate success when using short texts. With the use of contextual associations, researchers were able to improve the recognition of entities within Twitter texts (Jung, 2012). There are also other methods to increase performance in short texts entity recognition tasks. Using Bidirectional LSTM learning, research has shown that it is possible to have a decent working model to perform named entity recognition on short and noisy datasets without using hand-crafted features (Limsopatham & Collier, 2016). Research on Linking entities in shorter text databases has also shown some interesting findings. Using semantic relatedness between entities, researchers have shown that they can identify relationships between events and other types of entities. However, the framework used to detect relationship did seem to struggle with person entities. Different variations of a person were seen as unique entries, an example of this would be the found entity "Barack Obama" and "Mr. Obama" (Celik et al., 2011).

Within this research, we will concern ourselves primarily with the problem seen for entity linkage where multiple variations of one entity are being classified as unique entries. Detecting entities with many possible mention variations can be very useful. There are different reasons as to why a person might have different name mentions in texts. A very simple reason could be that someone made a mistake while typing, but there are also other reasons for variations in spelling. A lot of different variations can be found because of transliteration. A simple example of a difficulty found in transliteration is the transliteration of the Arabic letter ق , in simple terms pronounced as qāf. The pronunciation for the problematic letter varies from region and is different for many Arabic dialects. When transliterated to English it can be found as "g", "q" and "k". For only one simple Semitic letter we can already see three different ways of transliteration. The difficulty in transliteration for Semitic languages gives us good reason to believe that Arabic and other

Semitic names might have many different variations when transliterated to English, this makes them useful entities to use within our project. Transliterated texts containing an Arabic name are likely written by someone who's mother-tongue is Arabic. When people transliterate words from their mother-tongue to another language, they sometimes make mistakes because they try to mimic the sounds in their native tongue. However, people are not aware of any transliteration rules and might thus write names and words based on how they feel it should be written. Because Arabic is a highly morphological language, NER-tools for Arabic often have a harder time recognizing entities (Maloney & Niv, 1998). Not only is having a highly morphological language troublesome for Arabic NER-tools, but it could also potentially increase the number of mistakes people make by transliterating words and names. Other issues in transliteration can be found because of the type of Arabic that is being used. Arabic can be classified into three types: Classical Arabic, Modern Standard Arabic, and Colloquial Arabic Dialects (Elgibali, 2005). All forms of Arabic have their style of writing but are still very similar, however differentiating and having to distinguish which type of Arabic is being used, can be tricky for translators as it might not be their mother-tongue. Other Semitic languages might have similar characteristics and limitations which could increase the number of mistakes made in translations and transliterations. All the differences found between languages could potentially lead to a bigger amount of variations found for names transliterated from one language to another.

Finally, although there is a lack of information on linking entities without using pre-existing databases or encyclopedias, we can still look at research in which existing databases are used. Hachey et al. (2011) have shown that they can link entities in text with their corresponding Wikipedia page. Using Graph-based measures, the researchers have shown that they can create an unsupervised approach that can achieve an accuracy of 85.5%. They have shown that unsupervised approaches can be competitive with supervised approaches, indicating that unsupervised machine learning approaches are an effective method for linking entities. Other research done by Bunescu and Pasca (2006), shows that they can disambiguate entities using Wikipedia by detecting features of entities. The researchers have shown that Support Vector Machines (SVM) can improve results on named entity disambiguation. They have compared their results with the performance of a baseline model and showed promising results. Although SVM's are very popular in machine learning and artificial intelligence, they are not as popular as they used to be. There are some downsides to SVM's, therefore, it is important to examine the performance of other machine learning tools and approaches. As we can see from the researches done by Bunescu and Pasca (2006) and Hachey et al. (2011), linking entities and the disambiguation of entities is often done by using different encyclopedias or databases. Because we lack a pre-existing database or encyclopedia, to assist in linking our entities, we cannot directly compare the results from our research to other research. However, we can still see that machine learning tools show promising results in related fields.

3 Methodology

As we explained in our introduction, we have multiple goals that we try to tackle using different datasets and conditions. The different datasets and conditions we will use are discussed in more detail later on in section 3.1. For our main goal, we try to research how different models can classify which entity a short text contains. To give an example, we would like to know if our models could correctly classify an example sentence such as:

> *"He alleged that the Pakistan Army induced Usama bin Laden to lead an armed group of Sunni tribals."*

to contain the entity *"Osama bin Laden"*. We can see a different variation of the entity in the example than the entity we are searching for. However, we are still trying to correctly classify the entity even when a different variation is given. We will try to answer our main question by answering the three underlying research questions. To answer our first research question, in which we try to find out what influence the number of entities has on our research question, we reduce the number of entities we search for in our first dataset to five entities and test our models on this new dataset. We will also try to use our second dataset to see how lowering the number of entities influences the results of our models. Our second dataset contains at most 3 entity types, 2 entities sharing the same entity mention, and "unknown" entities. The second dataset we use will give us the option to search for 1 or 2 person entities. In conclusion, we will have the opportunity to see how our model will behave on a 10, 5, 2, and 1 entity search. To research our second question, we will apply three different conditions to our dataset, the conditions we have will alter the short texts slightly by either removing or replacing the entities sought for. We explain these three conditions in more detail in section 3.1.2. For our third research question, we have split our dataset into training and testing sets in three different ways, our splits will affect the number of variations on which we train and test. We will create different training and testing set splits using our first dataset containing 10 entities and its subset containing 5. The splitting methods we use are explained more thoroughly in section 3.1.1. In section 3.1.3 we describe our preprocessing phase that all our tweets have to go through before they are used by our models to classify the entities.

The three models we are using in our project are a Logistic Regression (LR) model, a Convolutional Neural Network (CNN) model, and a Topic Modeling (TM) model. We will give a brief overview of our models here, they will be discussed in more detail in sections 3.3, 3.4, and 3.5. We have also created some baseline models which use standard approaches of classification. The baseline models will be used to compare our model's performance to standard approaches of classification. Our baseline models are discussed in more detail in section 3.2. We will also use embedding techniques in our LR model and our CNN model. Looking at our model approaches, the first one we used is that of LR. Models using LR are known as logistic models. Logistic models can estimate the probability that an object belongs to a certain class or event (Tolles & Meurer, 2016). Simple logistic models are normally binary models, allowing them to classify an object as only one of the two modeled classes. Having a binary model can be a partial solution to our problem but will be problematic when searching for multiple entities. It is important to have the possibility of searching for multiple entities, this is because we also want to be able to differentiate between two very closely related or even similar names. Seeing as we want to be able to differentiate between entities we have, therefore, decided to use a multinomial LR approach. By taking a multinomial approach we can use the model not only for a binary problem but also for multi-class problems (Hensher et al., 2012). To achieve our goal and improve our model results, we use embedding tools based on the Word2Vec algorithm created by Le and Mikolov (2014). Our second model is the TM model, which tries to cluster our data. TM is a text-mining tool that is often used to discover hidden semantic structures in text bodies. TM can be used in multiple fields to find patterns in genetic data, images, and even social networks (Blei, 2012). Considering TM has a broad field of use, we intend to use it to help us cluster our short texts. By training a model with our dataset we will try and teach our model to cluster data based on our desired "topics". Our model is intended to cluster short texts based on the entity they contain, by doing this we can see if our model can cluster the name variations accordingly. The third and last modeling approach we will use to find entity variations is done with the use of a CNN. Neural Networks are a state-of-the-art tool often found within the area of Artificial intelligence. CNN's are used as tools to solve problems based on pattern recognition. Neural networks have seen impressive results in fields such as image recognition (Russakovsky et al., 2015), image classifications, medical image analysis, and natural language processing (Collobert & Weston, 2008). Because CNN's are a very up to date tool and strong identifiers for patterns, we will try and use these to help us solve the problem of finding name variations. We believe that certain patterns provide important information to classify a text and associate it with an entity. Our CNN will also be making use of a pre-trained embedding layer using vectors gathered with the help of the GloVe Algorithm. GloVe is an unsupervised learning algorithm, which can obtain word vector representations by mapping words into a meaningful space where the distance between words is related to semantic similarity (Abad et al., 2016). The pre-trained word representation are available on https://nlp.stanford.edu/projects/glove/ (Pennington et al., 2014).

3.1 Dataset

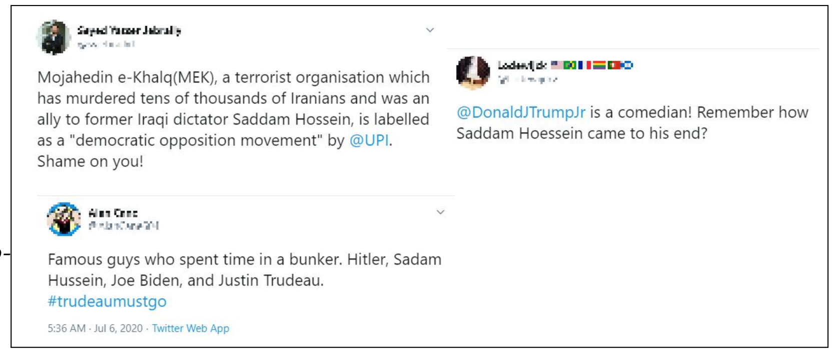
For our research, we have made use of a social networking and micro-blogging service known as Twitter. Users on the social networking platform can interact using messages known as tweets. Our datasets consist exclusively of tweets, these tweets have a maximum amount of 140 characters. Our first dataset, containing 10 unique entity, consists of 3874 unique tweets in total and has entities with easily distinguishable mentions. Tweets were gathered using the Twitter API and were searched for using hand-picked entities from the JRC database (Steinberger et al., 2013). The entities we gathered are well-known figures in middle-eastern politics. All our tweets consist of English tweets, which can be searched for with the Twitter API. The entity mentions in our tweets are all originally from a Semitic language but have been transliterated into the Latin script. Twitter users have mentioned the chosen entities in different contexts, but overall the settings in which they are found are mostly politically charged. The Twitter API has a limit on how far back tweets can be found, using it allowed us to find tweets that were made 7 days before the date of search. The Twitter API[1] also has a limit of 180 tweets each fifteen minutes, meaning we were able to obtain 17280 tweets a day in total. However, searching every 15 minutes is not a realistic option and would also be overkill seeing as we can only look back 1 week before the date of search. Finding 17280 extra tweets in a day would be unrealistic because some entities are not mentioned regularly. To get optimal results we spread our search queries over weeks and searched a couple of hours weekly. Therefore, most of our tweets are found around the months this project was started. The range of dates for the post times of all our tweets can be found from February 2020 to June 2020.

The most common entity mentions for each entity in our main dataset are: *"Mahmoud Ahmadinejad", "Ali Khamenei", "Benjamin Netanyahu", "Osama Bin Laden", "Saddam Hussein", "Recep Tayyip Erdogan", "Hosni Mubarak", "Yitzhak Rabin", "Abu Bakr", "Ehud Barak"*. We also have a dataset in which we only use a subsection of our 10 entities in which the amount of entities is

[1]For a better understanding of the Twitter API, https://developer.Twitter.com/en/docs shows all the needed information and details.

halved. The dataset uses only a subsection of the previous database and it contains the 5 names: *"Ali Khamenei", "Osama Bin Laden", "Saddam Hussein", "Recep Tayyip Erdogan", "Yitzhak Rabin"*

The JRC database also contains multiple varieties for entity mentions which are given by first and last name, there are many different forms found in the database. With the use of the JRC database, we were able to find 61 spelling variants in our tweets for the 10 entities we used, giving us a total of 71 variations. The number of variants found for an entity mention can differ for each entity. To give an example of found variations, we will use the person entity *"Abu Bakr"*. Abu Bakr has a total of 4 variations that can be found within our tweets. 1. The main entity mention *"Abu Bakr"* and 3 other entity mentions: *"Abou Bakr", "Abu Bakar", "Abu Baker"*. A full overview of all the variations that are found within our tweets can be found in the appendix in section A.1, the percentage of occurrence for each variation can be found as well. The overview of all the names found shows us that an entity can vary in both the first and last name. Seeing as our entity mentions consists of first and last names, we decided to query our search results using the full name mention variations only. To get an idea of how people mention entities within tweets we can look at figure 1. in which we can see 3 different variations of the entity "Saddam Hussein".


Mojahedin e-Khalq(MEK), a terrorist organisation which has murdered tens of thousands of Iranians and was an ally to former Iraqi dictator Saddam Hossein, is labelled as a "democratic opposition movement" by @UPI. Shame on you!

Famous guys who spent time in a bunker. Hitler, Sadam Hussein, Joe Biden, and Justin Trudeau. #trudeaumustgo

5:36 AM · Jul 6, 2020 · Twitter Web App

@DonaldJTrumpJr is a comedian! Remember how Saddam Hoessein came to his end?

Figure 1: Examples of entity mentions within tweets

We also have a second dataset, which consists of three unique entities. The entities found in our dataset all have the same popular Arabic *'Muhammad'* and 3 of its variations: *'Mohammed', 'Muhammed', "Mohammad"*. We will split our second dataset into 2 parts, we do this to look at how our models can perform on a single search or duo search. The first part consists of tweets mentioning the prophet Muhammad and other unknown Muhammads. The second part contains the football player Muhammad Salah and other unknown Muhammads. The full dataset will also be used containing the Prophet Muhammad, the football player Muhammad and unknown Muhammads. All of the dataset parts have the same four variations available for all the classes. Seeing as there are many different entities found within the two previously mentioned datasets, we have only labeled tweets as either "Prophet Muhammad", "Mohammad Salah" or "others". To gather this second dataset, we have applied the same techniques used on the first dataset. Our dataset consists of a total of 753 tweets, with an even distribution for the entity "Mohammad Salah", "Prophet Muhammad" and the unknown "Muhammads". In the collection of our dataset, we first queried the full entity names in the Twitter API, filtering retweets to eliminate duplicate results. By searching for the full entity mention on all variations, such as "Prophet Muhammad", we can then remove the "Prophet" part, leaving us with the remaining tweet containing only "Muhammad". We perform the same procedure for our entity "Muhammad Salah". For our unknown entities, we gathered our tweets by searching on all variations of "Muhammad". However, we manually went through the tweets until we collected enough tweets that we believed had no reference for our chosen entities.

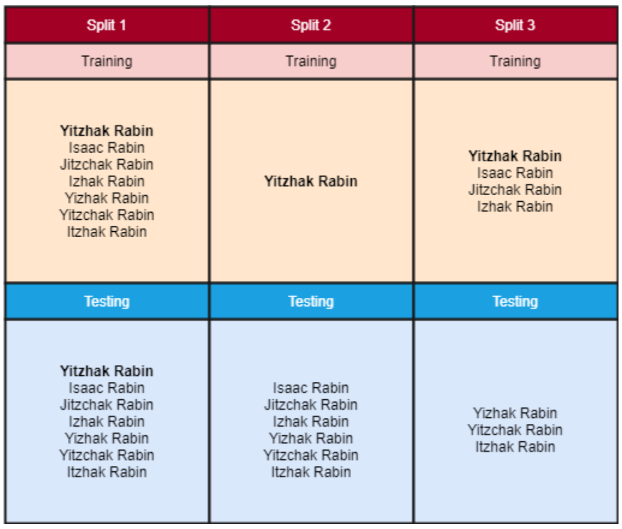
3.1.1 Dataset splits

To research how well our models can detect the variety of entity mentions, we have chosen to split our dataset in three different ways:

- by creating a training and testing set using a standard 80%-20% split using the main and all the variants.
- by creating a training set based on the entities' main name and by creating a testing set on its variations.
- by creating a training and testing set using data from the entities' main name. However, we will only use a part of its variants for training and some for testing.

The reason we decided to split our dataset in different ways is to see how well our models can perform when given different amounts of entity mention variations. We believe that the number of entity variations on which a model is trained might influence its performance when searching for all possible variations. Getting insight on how much training knowledge our models need to find all possible entity mentions, can help us research how well we can identify an entity within a tweet regardless of its mention variant.

For our splitting methods, we first start with a basic way of splitting our data in which we simply split all data using 80% for our training-set and 20% for our test-set. We intend to control the ratio for our main entity mention and each variant it has. Having an even split for all our names' data allows us to use our first dataset as a control set. It gives us the ability to see how well the model behaves when we have all knowledge of possible variant mentions available and train the model on them. We also have another splitting method, where we perform a standard split that learns training on only the main entity mentions and tests on finding all its variants. By splitting training and testing into main mentions and variants respectively, we want to see whether our models can perform using minimal variant knowledge. In our second dataset, the model has to detect other variants when we only know the context in which its main form is being used. For our final train and test set, we split our data using a part of its variants and its main name's data as a training set. All the unused data, that were not added to the training set, will be used as a test set. With the use of our third splitting method, our models will only be able to train using some variants. In our last splitting method, we will be able to see how well models train using more context and data, while not having complete information. The proposed methods will give us the ability to use our first split as a control method, our second split as a tool to discover new entities based on the main mention, and our last split gives us the ability to detect new variations with the help of a smaller number of mention variations. To get an overview of what the goal for each split would look like we refer to figure 2, where we can see what entity mentions are being used for training and testing.



Split 1	Split 2	Split 3
Training	Training	Training
Yitzhak Rabin Isaac Rabin Jitzchak Rabin Izhak Rabin Yizhak Rabin Yitzchak Rabin Itzhak Rabin	**Yitzhak Rabin**	**Yitzhak Rabin** Isaac Rabin Jitzchak Rabin Izhak Rabin
Testing	Testing	Testing
Yitzhak Rabin Isaac Rabin Jitzchak Rabin Izhak Rabin Yizhak Rabin Yitzchak Rabin Itzhak Rabin	Isaac Rabin Jitzchak Rabin Izhak Rabin Yizhak Rabin Yitzchak Rabin Itzhak Rabin	Yizhak Rabin Yitzchak Rabin Itzhak Rabin

Figure 2: Training and testing set examples acquired by splitting methods applied to entity mentions of Yitzhak Rabin, main entity mention is written in bold

For the Muhammad database task, we do not split them in the three ways explained above. In the Muhammad database, we are trying to see how well our models can differentiate entities, meaning that we don't need all the different splits for name variation detection.

3.1.2 Conditions

We have also created three special conditions for our databases to train and test our models. Using these three conditions, we want to see how well our models can detect entity mentions using only context knowledge. We apply three conditions to our dataset containing 10 different entities and its mention variations, and our dataset with 5 entities and all of its mention variations. The conditions will be applied to all three data splits mentioned above, meaning we will report 9 results per database in total. For both the mentioned databases, the three different conditions are:

- a standard condition: where the entity mentions and variations have not been adjusted
- a condition without entity mentions: in which the entities mentions have been removed
- a condition wherein entity mentions are replaced: where the entity mentions are replaced by the name *John*.

Because we are using embedding tools in some of our models, we think that having these conditions gives us valuable insights into our models' trustworthiness. Removing entity mentions in our tweets will leave us with only the remaining context. We already mentioned that most of our entities are found within political contexts, meaning that texts might be written using similar structures and even words. With the use of our embedding tools, we have geared our models towards a more semantically driven task instead of a trivial syntax task. Using our designed conditions, we believe that we can detect more clearly how well our models behave on a semantic-only task and the combination of both the semantic and syntactic tasks.

We use our standard as a control group allowing us to see how our models would behave when given full information on all entity mentions, here our models can use both semantic and syntactic information. We omit and replace entity mentions in our data to see what effect this has on our model results, giving us an indication of how important tweet contexts are for name variation detection. If no entity mention is found within a tweet, models are not able to detect variations based on syntactic similarity and will thus only be able to look at the semantic similarity of the tweets. Replacing names has a similar effect as removing the names, but has the potential to confuse the models and give us a slight insight into how well our models can differentiate given entities with the same person mention. We can find a visualization of our conditions applied to a single tweet in figure 3.

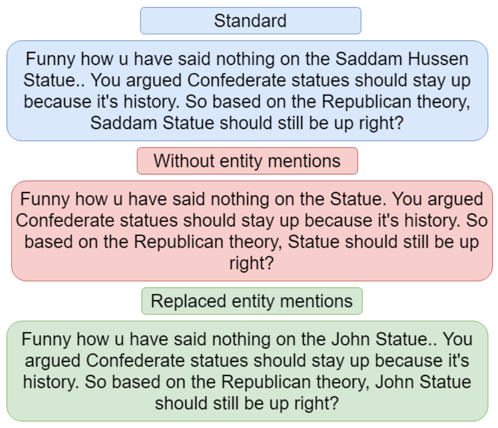

Standard
Funny how u have said nothing on the Saddam Hussen Statue.. You argued Confederate statues should stay up because it's history. So based on the Republican theory, Saddam Statue should still be up right?
Without entity mentions
Funny how u have said nothing on the Statue. You argued Confederate statues should stay up because it's history. So based on the Republican theory, Statue should still be up right?
Replaced entity mentions
Funny how u have said nothing on the John Statue.. You argued Confederate statues should stay up because it's history. So based on the Republican theory, John Statue should still be up right?

Figure 3: Example of conditions applied to a single tweet

We will also apply these conditions to the *"Muhammad"* dataset. By applying our conditions to our Muhammad dataset we don't expect very different behavior among them, as context should be very important seeing as all entities carry the same mentions. However, in our Muhammad dataset, we already know that there is a clear difference in context. Our entities are very different, one is a soccer player and the other is a religious figure. If our models rely too much on syntactic differences, they should not be able to distinguish our entities very well. A semantic approach is needed here as well to completely differentiate the entities within our given dataset. We made sure that the entity mention variants found within our Muhammad dataset are distributed evenly, giving we use embedding techniques and it might influence the importance of each variant.

3.1.3 Preprocessing

We have a preprocessing phase for all our tweets that is used for each method. The preprocessing phase first starts by turning everything into a lower case character. After that, we use a regex function to remove URLs. Next, we try and remove special characters such as @'s and #'s, as they add no substantial value for our research goal. Afterward, we will remove punctuation and we also remove leading and trailing white spaces from tweets. We then apply tokenization to our tweets and remove special characters such as newlines which can be found as tokens (`"\n"`). Then we look for stop-words in our tokenized list of tweets and we remove those if found. Next, we remove words that are only one character long, as they often hold no values. Finally, we stem all words using NLTK's PorterStemmer. Stemming has shown valuable results in multiple NLP tools and we believe it is more suitable for short text clustering (Le & Mikolov, 2014).

3.2 Baseline Models

We have implemented three different kinds of models in our research, however, we must check how well they perform when compared to existing models. Research on entity linking is often done using databases and encyclopedia to link entities in texts with. In our research we don't have any database or encyclopedia to link our entities with, this prompted us to design baseline models to get an indication of how successful our models are. We use three different types of baseline models; Bag-of-words based model, Levenshtein based model, and a key-matching model. Our first baseline model is based on a similar algorithm as our LR model. For our first baseline model, we will be using a Bag-of-words logistic regression model. Bag-of-words models are often used for classification based on texts, this allows us to compare the results of our models to that of the baseline model and determine whether our models have a better performance on all tasks (J. Wang et al., 2013). We will use the Bag-of-words algorithm to turn our texts into vectors and create a classifier based on our vectors. We create our vector using the Term Frequency Inverse Document Frequency (TF-IDF). TF-IDF is a common algorithm used to transform the text into a vector that is useful for machine learning tasks. To reduce the computation time of our model, we use a TF-IDF vector and limit our vector to a size of 400. Using a larger vector size might potentially improve results but at the cost of computation time. Testing larger vectors on a smaller dataset showed no significant increase in performance for our data. Because we have not seen any improvement in our small, we believe that a vector size of 400 is appropriate.

Aside from a Bag-of-words baseline model, we will also examine how well a simplified Levenshtein distance matching model would perform on our datasets. A Levenshtein model will compare strings given as input with strings specified by the user and determine whether the input string matches the specified string, based on the allowed edit distance. Our baseline Levenshtein model will be given entity mentions by us as input, instead of using an existing named entity recognizer, to see how well it would behave in optimal conditions. Our Levenshtein model will compare the entity mentions given as input to the main entity mentions available based on the edit distance we specify. Given a specific Levenshtein distance as maximum distance, we compare all variations of entity mentions to all the main entity mentions to determine how well our theorized model performs.

Performing our steps for the Levenshtein model will result in a table giving us scores that can be compared to the scores of our models. We will apply the Levenshtein algorithm on the main dataset with 10 entity mentions, in which we want to see how well we can discover new entities using the Levenshtein distance metric.

For our Muhammad dataset, instead of a Levenshtein matching model, we will use a model that performs key-matching using specific keys. A key-matching is a very simple rule-based model, that uses a preset of words to detect mentions of an entity. Doing key matching on our small entity detection test will enable us to research if our model can outperform a baseline model that differentiates entities based on target words. We need to see whether we can indeed outperform the baseline model for entity differentiation to ensure that our models do not perform the same procedure as our baseline key-matching model. It is possible that our model only looks at commonly found words as features, instead of the context and relationships. The keywords for our baseline model can be decided on in many ways. For example, we could look up our entities and decide on 5 words we believe are going to be commonly found around them or we could look at news articles about our entities and count common words that often appear in these texts. However, seeing as we have data that is from Twitter and we can only collect recent tweets, we have decided to take our keywords from Google Trends[2]. Google Trends analyzes the popularity of top search queries in Google searches across various regions and languages. Our baseline model will take the 5 most popular keywords taken from Google Trends, giving us a good indication as to what words are often surrounding our entities. We use the following five words for:

- The Prophet entity: *Islam, Muslim, Quran, Arabic, and Allah.*
- The Salah entity: *Liverpool, Liga, Premier League, Soccer, and Egypt.*

We intentionally left out some tags from Google Trends such as *Prophet* or *Salah* as we removed them from our tweets. The key-word matching model takes a tweet as input and will decide if it can label the tweet as one of our entities if a key-word is found. If no words are found, our baseline model will simply label the tweet as unknown. Similar models have been used for filtering out harmful content, such as pornographic content on the Internet (Su et al., 2004). Su et al. (2004) shows that they can create a fast and effective keyword matching model to classify pornographic content. Other researchers, such as Chaudhuri and Agrawal (2004), have shown that they were even able to find relational databases using preprocessing of data and keyword matching. Our task is also a classification task and thus we believe that a keyword matching model might show good results for our task as well.

[2]https://trends.Google.com/trends/?geo=US

3.3 Logistic Regression

In our LR model, we aim to classify our tweets based on the entity and its context that are found within the tweet. For our LR model, we need to take as input our tweets and have our classifier train on them. However, we will use embedding tools to increase the performance of our LR model. The embedding tools we use convert our data into vectors, this can be done in multiple ways. For our LR model, we decided to vectorize our data using Doc2Vec. Doc2Vec is an extension of Word2Vec, it uses an extra layer to create a vector representation for documents or paragraphs. The vector created by the extra layer in Doc2Vec is called the "paragraph vector". Le and Mikolov (2014) have shown that Doc2Vec is an effective tool for text representation and often outperforms the Word2Vec algorithm in text classification and analysis tasks. An added layer, that looks at document representation, is useful for our research seeing as each tweet can be taken as a paragraph. We use the Doc2Vec embedding tools, as we intend to look more at semantic information carried by our entities. In our research, we want to determine how well we can differentiate entities primarily by their semantic similarities. One further reason why we use Doc2Vec instead of other embedding tools, to convert our tweets into vectors, is because it creates vectors with lower dimensions. Having lower dimension vectors greatly reduces the computational load. Another reason to use word embedding is that they allow classifiers to incorporate the semantic property of words. Doc2Vec has 2 algorithms available to represent texts as vectors, the Distributed Memory (DM) model, and the Distributed Bag of Words (DBOW).

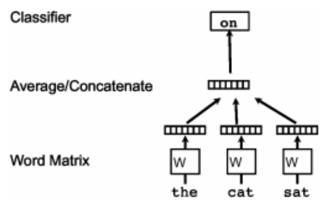


Figure 4: Visual representation of the CBOW model for word vector learning. The context of the three words *"the, cat, sat"* is used to predict the fourth word *"on"*.

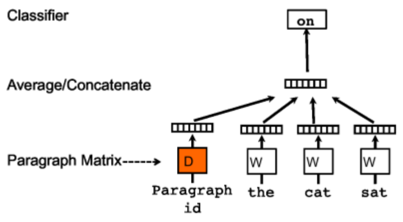


Figure 5: Visual representation of the DM model for word vector learning. The context of the three words *"the, cat, sat"* together with the paragraph vector is used to predict the fourth word *"on"*.

In our thesis, we will look at the performance of both algorithms separately as well as their performance when we combine them. The first model algorithm available for Doc2Vec is the DM model. The DM model is inspired by word vector learning methods that predict the next word in a sentence. However, the DM models' algorithm also creates a paragraph vector to help predict upcoming words in a sentence. It is similar to the Continuous-Bag-Of-Words (CBOW) algorithm from the Word2Vec model, which we have used in our baseline Bag-Of-Words model. A CBOW model predicts which word is most likely to occur within a context, it does this by finding the probability of a word occurring given the context of the surrounding words. As an example, when we have the sentence *"the cat sat [...]"* the CBOW algorithm will try to predict the word that needs to be on the empty spot. The algorithm does this by taking the contextual information given by the words *the, cat* and *sat* into account. For the DM model, the word and paragraph vectors are learned using a stochastic gradient descent which is obtained by backpropagation. A visualization of the frameworks for both the CBOW model and the DM model can be found in figures 4 and 5 (Le & Mikolov, 2014).

The second algorithm available in the Doc2Vec model is the DBOW model. The DBOW model is analogous to the skip-gram model in Word2Vec, which tries to predict the context when given one word. For example, when given a word such as cat our skip-gram model will understand that there is a huge probability that the context is: *"The [..] sat on"*. The model trains by randomly initializing and updating the document vector and its weights by using stochastic gradient descent via backpropagation. Unlike the DM model, the DBOW model predicts the context directly from the document vector. Predicting context from one vector allows encoding higher n-gram representations (Mani et al., 2018). A visual representation of the DBOW framework can be found in figure 6. Finally, we will use a combination of the two previous algorithms to see if this can improve the classification of our tweets. The combination of the algorithms has shown more robust results across many classification and analysis tasks and is highly recommended by the designers, Le and Mikolov (2014).

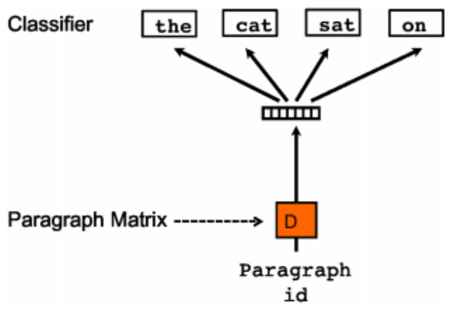


Figure 6: Visual representation of Distributed Bag-Of-Words model. The context of the words is predicted by using the document vector.

Using our word embedding tools we can create a vocabulary, which our model can use to classify our tweets based on entity context. To train the vocabularies of our models, we will initialize them using a dimensional vector length of 300. We will also specify that our models only use words that occur more than twice. Finally, the model will train using negative sampling, drawing 5 noise words. The vectors obtained from training our dataset using the algorithms can almost be used as features for our LR model. Although the vectors obtained from the algorithms are of appropriate dimension, we still need to standardize the values within the vectors. Machine learning processes often benefit from re-scaling data and, therefore, we decided to add this to our preprocessing phase. We use scaling tools to standardize our raw feature vectors independently into a representation that is more suitable for our LR model. After our preprocessing phase, we can train the LR model and test its accuracy on our test set. We will train the LR model on the vocabularies created by each of the previously discussed algorithms and their combination, resulting in 3 different LR models.

3.4 Short Text Topic Modeling

Short text topic modeling (STTM) is an emerging field within AI. Facebook and Twitter are very popular means of social communication and they both contain short texts. Research on short texts using long texts traditional methods such as Latent Dirichlet Allocation (LDA) and Probabilistic Latent Semantic Analysis (PLSA) might not perform as well on shorter text, seeing as they might have difficulties to find the right number of topics (Chen et al., 2011). Not being able to find the correct number of topics in a dataset with short texts can be problematic for models and thus create the need for new methods. Yin and Wang (2014) created a model that tries to cluster short texts, known as the Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture (GSDMM) model. The GSDMM model can deal better with sparse and high dimensional problems often found in short texts, showing that this type of model can be an interesting and potentially successful method for entity-based clustering. The Gibbs sampling algorithm is based on the LDA algorithm introduced by Blei et al. (2003), which uses prior knowledge to generate documents made of topics and updates them accordingly. A mixture model is a probabilistic model, used to represent the presence of segments within an overall population. The Gibbs sampling approach assumes that each document only contains one topic and will, therefore, create documents using the same unique topic and not based on a mixture of topics as is done in the LDA model.

The Dirichlet Multinomial Mixture (DMM) part of the algorithm is a probabilistic generative model that assumes documents (in our case tweets) are generated by a mixture model and that there is a one-to-one correspondence between the components of the mixtures and the clusters (Yin & Wang, 2014). The DMM model generates the likelihood of the document existing by looking at the sum of its total probability over all clusters. In figure 7 we can see how the initialization parameters influence the DMM's probabilities. The probability of a document being generated is dependent on the cluster weights. Afterward, the model is generated according to the cluster it belongs to.

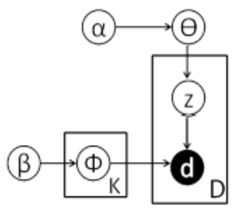


Figure 7: Visual representation of the Dirichlet Multinomial Mixture (DMM) Model. α and β represent initial parameters that influence the weights. K represents the models generated mixture components (clusters). D represents all documents in the corpus and d a single document. z represents a single cluster.

The algorithm for the GSDMM derives from the DMM model and starts with a random assignment of documents to the chosen amount of clusters. After the random assignment of documents, the information on cluster labels for each document is recorded. The information that is recorded is the number of documents, the number of words, and the number of occurrences of a word in a cluster. After initialization, each document, in turn, gets his cluster re-assigned according to a conditional distribution in which its labels are removed from the cluster. The re-assignment of the clusters step is repeated 30 times in our case. Each iteration the recorded information gets updated, leaving only a part of the initial clusters at the end of the procedure. The GSDMM model uses multiple parameters in the model's equation which have to be set beforehand, they relate to the prior probabilities of documents to be assigned to a cluster and the importance of the clusters' content towards the allocation of new documents. Two of the parameters in the GSDMM model are known as α and β. For the GSDMM model to work one has to assume the values, based on expected behavior. We decided to take an iterative approach to decide the values for the parameters α and β. For our project we test our model using steps of 0.1 in a range from 0 to 1, these are the maximum values that fit the given equations of the algorithm. By taking an iterative approach we can check multiple values and take the ones that perform best for our research. Seeing as speed is one of the strengths of the algorithm, we do not have to worry about our iterative approach being very time-consuming. The other and last parameter that is used in the model's equation is the number of expected clusters. The model doesn't exactly need to know how many clusters are needed, it only needs an upper limit. The parameter for the number of expected clusters can be assigned any number higher than what is expected because the algorithm will create empty clusters and find the exact number of clusters that are needed if working as intended. For our data, we will set the appropriate amount of clusters based on the entities found. Choosing a higher amount of clusters will not benefit performance and will only be computationally more demanding. After training models with chosen parameters, the model can be tested and accuracy can be calculated.

3.5 Convolutional Neural Network

For our CNN model, preprocessing the data as explained in section 3.1.3 is only the first step for our input data. After preprocessing our data we use a pre-trained embedded dataset obtained by using GloVe. GloVe is an unsupervised learning algorithm for obtaining words in their vector representation. The training for GloVe is done on aggregated global word-word co-occurrence statistics from the corpus (Pennington et al., 2014). Using pre-trained word embeddings, instead of creating our own, allows classifiers and models to recognize similar words without having been trained on them. Another reason why we do not create our own Glove embedding dataset is because our dataset is not large enough. Using a small dataset to create our own embedding might not find semantic relationships between words. The reason why a small dataset won't find semantic relationships between words is due to the small occurrences of words. If more data is available, words with similar meanings will occur more often in similar contexts. For the aforementioned reasons, we believe that using pre-trained GloVe embeddings will increase the performance of our CNN model. The use of embedding tools is needed for our CNN model because our data is too large to represent as simple one-hot encoded vectors. We could try to limit our vectors as we have done in our baseline Bag-of-words model. However, limiting our vector range would mean that information is lost, which can lead to a decrease in performance.

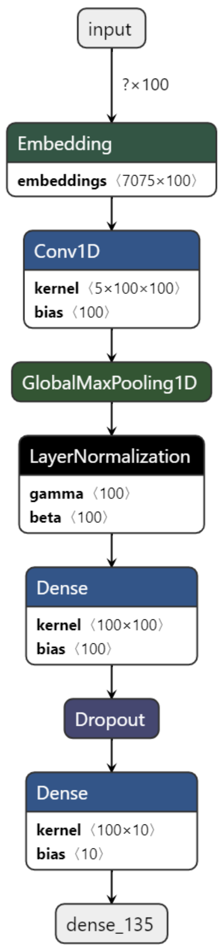


Figure 8: CNN architecture

Our CNN model starts with an embedding layer that transforms our discrete and sparse 1-hot vectors into a continuous and dense latent space. Then our model sends its embedded vectors to a simple 1-dimensional convolutional layer, with a Rectified Linear Unit (ReLU) activation function. After having our input data being convolved, our input data is then sent to a pooling stage. In our pooling stage, we down-sample our data with a max-pooling approach. Then our next operation is to normalize our data which is done in our normalization layer. Afterward, we send our data to our first hidden dense layer with 100 neurons and a sigmoid activation function. We choose a hidden layer with 100 neurons because it will allow our model to find enough node relationships for 10 different entity mentions. Then we apply a dropout layer to help us avoid overfitting. Finally, we send our data to our last layer containing a certain amount of neurons, respective to our dataset, and a softmax activation function. An example and overview of our CNN its architecture can be found in figure 8. Figure 8 shows how our CNN works for a dataset of 10 entities. We compile the model using an ADAM optimizer with a learning rate of 0.0001 and using categorical cross-entropy as our loss function. Our learning rate was analyzed using trial and error, we validated our result outputs by training and validating on our datasets and looking at what would be an acceptable rate. Cross-entropy is recommended and often used for multi-class classification tasks. We fit our model using our training data in 10 epochs with a batch size set to 6, meaning that each epoch we will use 1/6th of the total dataset. Taking 1/6th of the dataset for each epoch allows us to train and test on different entities and tweets each epoch. For example, our CNN model using the dataset with 10 entity mention will be training and testing on 646 tweets each epoch. The amount trained and tested on will be dependent on the split chosen. By training on different data each epoch, the CNN model will perform better as it reduces the error of training data. We chose 10 epochs after analyzing multiple different options. Using 10 epochs reduced the number of errors, while also not over-fitting the data.

4 Results

Before we report the results for all our models, we report how well our baseline models perform on our datasets. Our baseline models will be tested on the same datasets our main models will be using. The baseline model results will give us a benchmark that we are aiming to improve with our main models. Afterward, We will start with the results for our main dataset containing 10 political figures, we then show our results for the sub-selection that contains 5 entities. Finally, we will show our results for our models using our Muhammad dataset. For each section, we will report the results gathered from each model. In our result section, we will try to keep our tables minimal to sustain oversight of our gathered results, giving us only a broad view of our gathered results. We will however mostly refer to tables and figures in our appendix, as these are needed to understand the significance of our gathered outcomes.

4.1 Results for Baseline models

For our baseline Bag-of-Words logistic regression model, we see that we are only able to achieve a top accuracy of **36,9%** when tested using our 10 entities dataset. Table 1 shows us the results for each split. Although the accuracy achieved is not very high it is still way higher than having a model that would classify at random. A random classifier would average an accuracy score of 10%, Seeing as there are 10 names. We see that in split 1 the best performing condition is the replaced condition, however, none of the differences found in the results for split 1 are significant ($p > 0.05$). In split 2 we see that there is a significant difference between the standard condition compared to the without condition ($p = 0.04$) and the standard condition compared to the replaced condition ($p = 0.007$). In split 3 we again see that the replaced condition is the best performer, nonetheless, the difference found here is again non-significant.

Table 1: Precision, Recall, F1 and Accuracy results for baseline Bag-of-Words model trained on dataset using 10 entities for each split under 3 different conditions.

		Precision	**Recall**	**F1**	**Accuracy**
Split 1	**Standard**	0.356	0.358	0.315	0.358
	Without	0.149	0.164	0.152	0.164
	Replaced	0.368	0.369	0.359	0.369
Split 2	**Standard**	0.529	0.252	0.286	0.252
	Without	0.214	0.138	0.158	0.138
	Replaced	0.246	0.164	0.181	0.164
Split 3	**Standard**	0.232	0.120	0.138	0.120
	Without	0.180	0.158	0.160	0.158
	Replaced	0.232	0.174	0.190	0.174

In table 2 our results using 5 entities are shown. We see that for our first split we can achieve similar scores on each condition. The results achieved by our baseline model on the first split do not significantly differ from each other. In our second split we see that our baseline model performs significantly worse in our standard condition compared to both the without condition ($p = 0.02$) and the replaced condition ($p > 0.01$). In our third split for 5 entities, we cannot find any significant differences even though we see a very high performance for our standard condition.

Table 2: Accuracy and F1 score results for baseline Bag-of-Words model trained on dataset using 5 entities under different conditions.

		Precision	**Recall**	**F1**	**Accuracy**
Split 1	**Standard**	0.386	0.305	0.267	0.305
	Without	0.217	0.226	0.217	0.226
	Replaced	0.335	0.347	0.339	0.347
Split 2	**Standard**	0.205	0.051	0.049	0.051
	Without	0.245	0.223	0.214	0.223
	Replaced	0.351	0.308	0.312	0.308
Split 3	**Standard**	0.591	0.596	0.560	0.596
	Without	0.317	0.294	0.297	0.294
	Replaced	0.402	0.367	0.363	0.367

For our Muhammad datasets, we can see that when searching for the entity Salah our model can detect a mention of this entity with **66,7%** accuracy. Table 3 shows that this accuracy is significantly higher than the standard condition, $p = < 0.001$. When identifying the prophet Muhammad, we see that our standard dataset is the best performer. Although our standard dataset has the highest accuracy and seems to be working better than when removing entity mentions, we do not see any statistically significant differences between the standard condition and the without entity mentions condition ($p = 0.38$). Between our replaced dataset and our dataset, however, there is a significant difference found ($p = 0.008$). When looking for both the prophet and Salah, we again see that our standard condition is the best performer. But when looking at statistical significance, we cannot infer that any difference found between conditions is significant ($p > 0.05$).

Table 3: Accuracy and F1 score results for baseline Bag-of-Words model trained on Muhammad datasets.

		Accuracy	**F1**
Salah	**standard**	0.540	0.449
	without	**0.667**	0.659
	replaced	0.603	0.581
Prophet	**standard**	**0.619**	0.583
	without	0.524	0.501
	replaced	0.603	0.581
Salah & Prophet	**standard**	**0.400**	0.355
	without	0.337	0.327
	replaced	0.368	0.370

For our Levenshtein model, table 4 shows us how well a Levenshtein distance-based model would perform given the perfect scenario. With the perfect scenario, we mean that our model will only compare entity mentions found in tweets. The entity mentions in our perfect scenario are given by us and not found by the model itself. Since the 5 entity dataset is a subset of our main dataset we do not expect much difference and we, therefore, don't believe creating a model for this is any use. The results are based on measuring the full name main mentions of every entity with every variation of all entity mentions, using the given distance. For example, Abu Bakr will not only have its distance measured with its own variants but also of the variants of Ali Khamenei, Benjamin Netanyahu, and all others. The procedure is repeated for all main names and the accuracy, precision, recall and F1 score are then calculated using the total correctly classified names. The results found using our Levenshtein model show that precision is very high at the start and gets lower the larger our distance becomes. For recall we see the opposite of what we see for precision, our recall starts lower and ends up at its maximum when we reach our maximum distance. It is important to take notice of our recall and precision seeing as our accuracy seems to always be high, but does not include how our model is getting results we do not want. A recall of 50%, which is found in our model using distance 1, indicates that we have only found half of all the relevant items. When we tune our model to find all relevant items, we have a precision of 50,7%. The precision shows us that we only achieved a recall of 100% because we have classified many items that were not relevant. The best result found is when we set our distance to 7. We have a balanced recall and precision and good F1 and accuracy scores. Although we can find good results using the Levenshtein method, we have to keep in mind that a model based on Levenshtein can not even be close to realistic or useful unless a perfect named entity recognition tool exists. In our project we do not mention the position of our entities, we only give a raw tweet as the input making Levenshtein distance hard to use. Other edit distance metrics, which we explained in section 2, will show similar problems even if they would have higher scores. However, our Levenshtein results are a good indicator of how well new variations of entities can and should be found.

Table 4: Accuracy((TP+TN)/(TP+FP+FN+TN)), True Positive Accuracy (TP/TP+FP+FN), Precision, Recall and F1 score results for Levenshtein model based on maximum Levenshtein distance given perfect scenario where main entity mentions are compared with all variations. The total scores are given using all main entity mentions.

Levensteihn Distance	**Accuracy**	**True Positive Accuracy**	**Precision**	**Recall**	**F1**
0	0.958	0.577	1.000	0.577	0.732
1	0.969	0.690	1.000	0.690	0.817
2	0.972	0.718	1.000	0.718	0.836
3	0.973	0.732	1.000	0.732	0.846
4	0.972	0.722	0.981	0.732	0.839
5	0.972	0.737	0.918	0.789	0.848
6	0.973	0.774	0.833	0.915	0.872
7	0.962	0.719	0.734	0.972	0.836
8	0.944	0.640	0.640	1.000	0.780
9	0.879	0.452	0.452	1.000	0.623

There are more problems found when trying to use a model based on Levenshtein distance. The Levenshtein model cannot use context which can be a problem for our research. We want to see if we can detect entities based on context only in our task where we omit entity mentions. Removing the entity mentions or replacing them will leave our model with 0% accuracy because no entity can be found in the text. Due to our model not being able to use the context of tweets, we encounter another problem. When using a Levenshtein baseline model to discern ambiguous mentions, we see that our model is unable to perform the task. To see how ambiguity can be an issue for our Levenshtein model we can look at our condition in which we replace all names or we can look at our Muhammad dataset. For our Muhammad dataset, we try and look for the correct Muhammad among multiple Muhammads and variations. Seeing as our main entity uses all 4 variations of Muhammad and all our unknown entities also share similar entity mentions, a Levenshtein model will never be able to correctly classify the correct entity based on name only. A Model-based on Levenshtein will be able to find all the Muhammads that are about one entity but will also have lots of false positives. An example of a Levenshtein model tested on one of the entities is found in table 5.

Table 5: Accuracy,Precision, Recall and F1 score results for Levenshtein model given perfect scenario where only entity mentions are compared. Tested on Muhammad dataset with maximum distance of 2.

	Accuracy	**Precision**	**Recall**	**F1**
Prophet	0.500	0.500	1.000	0.667
Salah	0.500	0.500	1.000	0.667
Prophet & Salah	0.330	0.330	1.000	0.496

Seeing as the Levenshtein model results are not a very good indication for a baseline, we have also created a baseline model based on key matching. The results for our baseline model using key matching can be found in table 6. Our key-matching model used 5 words for each entity which we explained in section 3.2. Because of the use of keywords words for our key-matching model, the results will not be affected when the entity mention is removed or replaced. The results indicate that some entities can be detected with high accuracy, however, the effort that is needed to be put in for each entity might be too intensive. The keywords have to be initialized for each entity and if a large number of entities need to be classified in a short text, then the amount of time searching for keywords might become an issue. Nonetheless, one could improve and solve this problem by automatically find keywords but this might still not be sufficient. Aside from the initial time investment needed for the initialization of a keyword-matching model, it might not be as effective for other entities as they could have very similar keywords. In our example, we chose entities with very distinct backgrounds and contexts allowing a keyword matching model to thrive.

Table 6: Accuracy and F1 score results for Key-matching model, tested on Muhammad dataset using 5 keys. Keys used in the model represent 5 most common words found with entity on Google Trends.

	Accuracy	**F1**
Prophet	0.694	0.766
Salah	0.741	0.851
Prophet & Salah	0.510	0.510

4.2 Results for 10 entities

4.2.1 Linear Regression Model

We measured the accuracy and the F1 scores for all our LR models. Before we discuss all our results in detail, we will start by only looking at the results for the best performing model. Our best performing model is the combined model, Table 7 shows how well the combined model performs on each split for each condition. we see that the standard condition is the best performer overall with barely any differences between the without entity mentions condition and the replaced entity mentions condition. We can also see that in split 1 we heavily outperform the second and third split options.

Table 7: Precision, recall, F1 score and accuracy for Linear Regression model using 10 entities for the standard, without and replaced conditions. Values are measured using the combination of the Distributed Memory model and the Distributed Bag of Words model.

	Split 1			Split 2			Split 3		
	Stand.	With.	Repl.	Stand.	With.	Repl.	Stand.	With.	Repl.
Precision	**0.588**	0.469	0.444	**0.347**	0.306	0.273	**0.332**	0.311	0.309
Recall	**0.590**	0.473	0.442	**0.421**	0.362	0.311	**0.341**	0.305	0.337
F1	**0.586**	0.464	0.441	**0.339**	0.295	0.249	**0.319**	0.299	0.297
Accuracy	**0.609**	0.488	0.450	**0.393**	0.364	0.311	**0.438**	0.391	0.385

To show how we got to the conclusion that our Combined model is the best performer, we must take a more in-depth look into all our results found in the tables in the appendix. In table 33 in our appendix, when looking at the dataset under normal conditions, we see that the DBOW model and the Combination of models have similar scores. The differences found between the DBOW model and the combined model are not significant, $p = 0.515$. The combination of models scores significantly better than the DM model on the first split, this is similar for the DBOW model when compared with the DM model (p <0.001). When looking at the second split, we can see similar results as in the first split. Both DBOW and the combined model score significantly better than the DM model ($p < 0.001$), but the differences between DBOW and the combined model are not significant ($p = 0.832$). On the third split, however, we see that a combination of both the models performs best. The difference found in the third split is not as big as before, nonetheless, the difference between the combination of models and the DBOW model is clearly significant ($p < 0.001$). Nonetheless, the differences between the combination of models and the DM model is not as significant ($p = 0.12$). The highest accuracy found overall splits is that of the combination of models in the first split, **60,9%**. From the significant differences found in the standard condition, we can thus conclude that the combined model seems to at least outperform one of the other models for each split.

When removing the entity mentions, table 34 in our appendix shows us that the combination of both models always outperforms the single models. In the first split, we can see that these differences are significant, $p < 0.001$. The highest accuracy found here is the highest accuracy for our dataset without entity mentions, **48,8%**. When looking at split 2, we see that a combination outperforms the other 2 it is only by a slight margin and that is reflected in the statistical significance between the models (p >0.05). A similar pattern is found in split 3, in which we also cannot take the differences found as statistically significant. For the DBOW model and the Combination, the results achieved by the model using the datasets without entity mentions on the first split are significantly lower than the dataset with entity mentions (p <0.001). The DM model is barely different and is found to be insignificant ($p = 0.75$). We can also see a decrease in performance for DBOW in the second split when using the dataset without entity mentions($p = 0.002$). The differences for the third split are all insignificant, $p > 0.05$. The results from our second condition again show a consistently good performance from the combined model, making it the best performer on average.

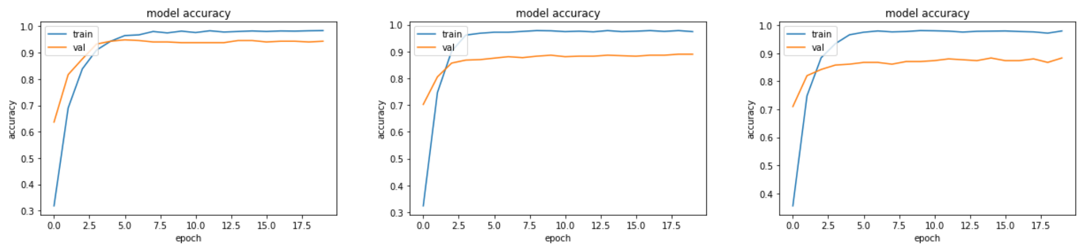
Replacing all the names shows almost the same pattern of differences as in the without entity mentions conditions, the results can be found in our appendix in table 35. The significance of the difference between splits and models are the same as described above. We see significant differences (p <0.001) for the DBOW model and the combination of both models in splits 1 and 2, compared to the dataset with entity mentions. For split 3 without entity mentions, there are no significant differences found when compared to split 3 with entity mentions. The highest accuracy we can find in our dataset with replaced entity mentions is a non-significant difference ($p = 0.146$) compared with the accuracy of the dataset without entity mentions, **45,0%**. In our last condition, we can again conclude that the combined model is the best performer given its results seen on all splits.

4.2.2 Convolutional Neural Network Model

We were able to gather the data for our CNN model for all our datasets. In table 8 we show the results for our dataset under the standard condition. When looking at the validation scores, we can see that the model performs best using split 1, where our CNN model has **94,3%** accuracy. From our table, we are also able to find that split 2 has a slightly higher accuracy score on the validation set compared to split 3. However, we see that split 3 has a higher F1 score and lower loss compared to split 2. When looking at the validation accuracy over time, we see that in split 1 our CNN model is only slightly affected by overfitting. The other two splits, however, are affected more by overfitting. Both splits 2 and 3 show an almost 10% overfitting which can be seen in the graphs in figure 9. To get a complete overview of the loss and accuracy we can look at table 24 in the appendix. The results found in the appendix show that the loss has a similar trend as seen in the accuracy for each split.

Table 8: Final training and validation loss, accuracy and F1 score after 20 epochs given per split for CNN model using 10 entity dataset that is under *standard* condition.

	Split 1		Split 2		Split 3	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.061	**0.180**	0.067	0.462	0.056	0.454
Accuracy	0.984	**0.943**	0.974	0.890	0.980	0.883
F1	0.983	**0.944**	0.974	0.874	0.980	0.881



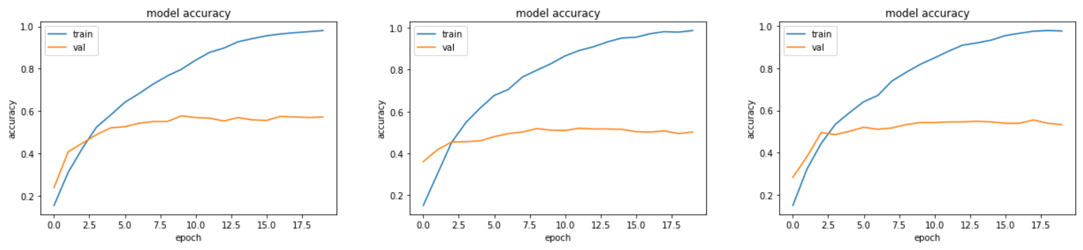
(a) Split 1 accuracy (b) Split 2 accuracy (c) Split 3 accuracy

Figure 9: Training and Validation accuracy for CNN model using dataset with 10 entities under standard condition. Results are shown per split over the duration of 20 epochs.

For the other two conditions where our entity mentions are either replaced or omitted, we can find similar success in classification when using the first splitting method. The results in table 9 show a maximum accuracy of **57.1%**, found on our validation set using split 1. The accuracy found for the condition where entity mentions are omitted is significantly smaller than the standard condition, $p < 0.001$. In our second condition, where entity mentions are omitted, we experience overfitting in all our splits which is visible in figure 10. Our loss is also significantly higher than in the previous condition for both training and validation which can be found in the appendix table 25.

Table 9: Final training and validation loss, accuracy and F1 score after 20 epochs given per split for CNN model using 10 entity dataset that is under the *without entity mentions* condition.

	Split 1		Split 2		Split 3	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.227	**1.309**	0.214	1.551	0.216	1.421
Accuracy	0.980	**0.571**	0.987	0.502	0.976	0.533
F1	0.969	**0.558**	0.973	0.434	0.972	0.508



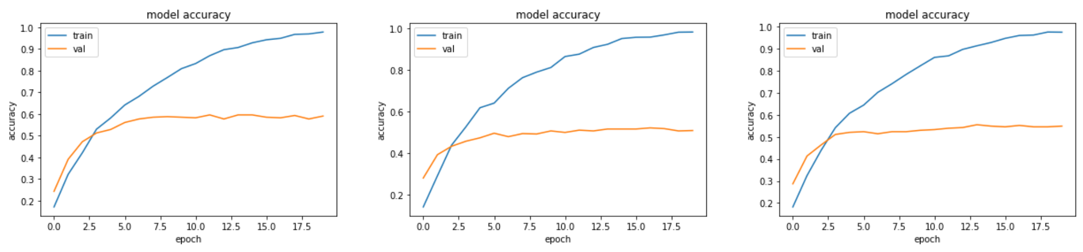
(a) Split 1 accuracy (b) Split 2 accuracy (c) Split 3 accuracy

Figure 10: Training and Validation loss/accuracy for CNN model using dataset with 10 entities under the *without entity mentions* condition. Results are shown per split over the duration of 20 epochs .

For our third condition, found in table 10, our highest accuracy can be found when we look at split 1. We see an accuracy of **59.0%** , which is a non significant difference from the dataset without entity mentions ($p = 0.45$). However, the difference between the replaced entity mentions dataset and the standard dataset is significant, $p < 0.001$. Figure 26 shows us similar overfitting as seen in our condition without entity mentions. The graphs for the accuracy in figure 11 show us that most of the splits carry a big amount of overfitting of almost 50%.

Table 10: Final training and validation loss, accuracy and F1 score after 20 epochs given per split for CNN model using 10 entity dataset that is under the *replaced entity mentions* condition.

	Split 1		Split 2		Split 3	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.243	**1.212**	0.232	1.537	0.227	1.384
Accuracy	0.978	**0.590**	0.982	0.509	0.975	0.549
F1	0.968	**0.567**	0.968	0.420	0.970	0.509



(a) Split 1 accuracy (b) Split 2 accuracy (c) Split 3 accuracy

Figure 11: Training and Validation loss/accuracy for CNN model using dataset with 10 entities under the *replaced entity mentions* condition. Results are shown per split over the duration of 20 epochs.

4.2.3 Gibbs Sampling Dirichlet Multinomial Mixture Model

The results for the GSDMM model using the standard dataset can be found in table 11. The results for our GSDMM model show that, when using the standard dataset with entity mentions, our third split performs negligibly better than our other splits. The highest accuracy found for our standard dataset is **80,1%**.

Table 11: Accuracy, precision, recall, F1 score and parameter settings given per split for GSDMM using 10 entities dataset that is under *standard* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Split 1	Split 2	Split 3
Accuracy	0.774	0.800	**0.801**
Precision	0.724	0.773	**0.801**
Recall	0.774	0.800	**0.801**
F1	0.722	0.781	**0.787**
α	0.7	0.5	0.4
β	0.4	0.9	0.4

When looking at the results for our dataset without entity mentions in table 12, we can see that our performance has drastically decreased. The best accuracy we can reach is around **50,9%**. The difference in accuracy is almost 30%, which is found to be a significant difference (p <0.001).

Table 12: Accuracy, precision, recall, F1 score and parameter settings given per split for GSDMM using 10 entities dataset that is under the *without entity mentions* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Split 1	Split 2	Split 3
Accuracy	**0.509**	0.454	0.508
Precision	0.533	0.500	**0.545**
Recall	**0.509**	0.454	0.508
F1	**0.505**	0.391	0.487
α	0.5	0.7	0.5
β	0.4	0.8	0.4

When replacing the entity mentions, we see in table 13 that our results decrease even more than when using the dataset without entity mentions ($p = 0.01$). The highest accuracy which can now be found is only **47,9%**. The overall decrease from the dataset with entity mentions to that of the dataset with replaced entity mentions is a significant one, $p < 0.001$.

Table 13: Accuracy, precision, recall, F1 score and parameter settings given per split for GSDMM using 10 entities dataset that is under the *replaced entity mentions* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Split 1	Split 2	Split 3
Accuracy	0.472	0.478	**0.479**
Precision	**0.573**	0.404	0.534
Recall	0.472	0.478	**0.479**
F1	0.419	0.415	**0.490**
α	0.9	0.4	0.1
β	0.5	0.5	0.2

4.3 Results for 5 entities

4.3.1 Linear Regression Model

For our dataset with 5 entities, we see that our combined model is again the best performer on most tests. In table 14 we can see how well our combined model performs using each condition on each split. compared to the first dataset using 10 entities, we see that in the first split our model scores better when under the replaced entity mentions condition. In the other 2 splits, we see that the standard condition is the best performer. The other 2 conditions seem to be both equally worse than the standard condition.

Table 14: Precision, recall, F1 score and accuracy for Linear Regression model using 10 entities for the standard, without and replaced conditions. Values are measured using the combination of the Distributed Memory model and the Distributed Bag of Words model.

	Split 1			Split 2			Split 3		
	Stand.	With.	Repl.	Stand.	With.	Repl.	Stand.	With.	Repl.
Precision	0.594	0.589	**0.631**	**0.542**	0.444	0.466	**0.632**	0.565	0.530
Recall	0.583	0.583	**0.640**	**0.631**	0.509	0.549	**0.712**	0.629	0.579
F1	0.585	0.580	**0.628**	**0.535**	0.418	0.446	**0.637**	0.570	0.531
Accuracy	0.577	0.586	**0.628**	**0.526**	0.430	0.467	**0.641**	0.571	0.555

Our results for the combined model found in the table above was gathered by looking at all the tables in our appendix using the 5 entity dataset. The results above are only a broad overview of our LR model's performance, the following part will go more in-depth for each result found in our appendix. Table 36 in our appendix shows us the results using the dataset with 5 entities under the standard condition. From our results in table 36 we can see that the highest accuracy in split 1 is **60.7%** and it is achieved by the DM model. The highest accuracy found in split 1 by the DM model is not significantly different ($p = 0.392$) from the combination of models but it is significantly higher than that of the DBOW model ($p < 0.001$). In our second split, we see that our models perform slightly worse than how they did in our first splits. The highest accuracy achieved we can find in our second split is **56,3%**. We see that the DBOW model and the combination of models behave similarly for the second split and that both of them are significantly different from the DM model ($p < 0.001$). When looking at split 3 we see that the combination model performs significantly better than the DM model, $p = 0.005$. Even though the DBOW model might be performing worse than the combination of models, the difference does not seem to be significant ($p = 0.267$). With an accuracy of **64,1%**, split 3 is the best performing split for our dataset with entity mentions. The accuracy found in split three by our combined model is only significantly better than the performance of the DBOW model in split 2, but not better than the performance of the DM model in split 1 ($p = 0.597$).

Looking at split 1 in table 37 for our dataset without entity mentions we notice that our highest accuracy does not drastically differ from the dataset with entity mentions. The highest accuracy we can find without entity mentions in our dataset in split 1 is **58,6%**, this is insignificantly different from the dataset with entity mentions ($p = 0.597$). For all our models without entity mentions in split 1, we can see that the combined model and the DM model perform similarly and both outperform the DBOW model significantly (p < 0.001). For split 2 in our dataset without entity mentions, we see that all models have no significant differences ($p < 0.001$). When comparing split 2 without entity mentions and split 2 with entity mentions, we see that without entity mentions the best performing model is significantly worse ($p < 0.001$) than the best model with entity mentions. In our third split, we can see that the combined model and DM model have similar results and both are significantly different from the DBOW model ($p < 0.001$).

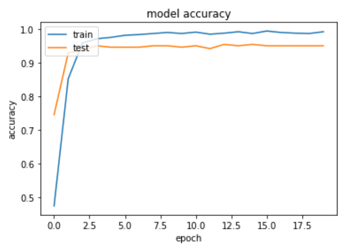
In table 38 we see the results for our models using the dataset with replaced entity mentions. In split 1 we barely see any significant differences compared to the results for the dataset without entity mentions. The combined model in split 1 has the highest accuracy from all splits and is significantly higher than the DBOW and DM models ($p < 0.001$). The results from the combined model in split 1 with replaced entity mentions, is not significantly different from the best result achieved by the dataset without entity mentions in table 37 ($p = 0.182$). On the other two splits, the combined model is significantly better than all other models, p <0.05. The differences found on the dataset without entity mentions and replaced entity mentions are mostly insignificant for splits 2 and 3.

4.3.2 Convolutional Neural Network Model

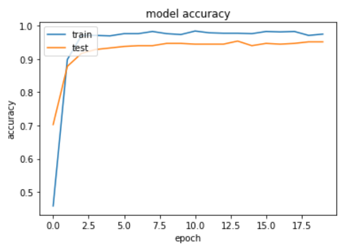
When we look at our CNN model using 5 entities, we see that it performs slightly better than when using 10 entities. Table 15 shows that the dataset with 5 entities under standard condition achieves **96,2%** accuracy. The increase in performance is seen on all 3 splits. When we look at the graphs in figure 27, we see that the overfitting has reduced slightly compared to the dataset with 10 entities.

Table 15: Final training and validation loss, accuracy and F1 score after 20 epochs given per split for CNN model using 5 entity dataset that is under *standard* condition.

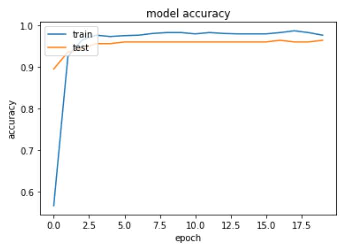
	Split 1		Split 2		Split 3	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.035	**0.113**	0.050	0.234	0.040	0.176
Accuracy	0.986	**0.962**	0.980	0.931	0.985	0.955
F1	0.986	**0.962**	0.981	0.925	0.986	0.947



(a) Split 1 accuracy



(b) Split 2 accuracy



(c) Split 3 accuracy

Figure 12: Training and Validation accuracy for CNN model using dataset with 5 entities under standard condition. Results are shown per split over the duration of 20 epochs.

Our second condition without entity mentions, when looking at table 16, shows that the CNN model can achieve an accuracy of **70,5%**. The accuracy achieved in our dataset without entity mentions is significantly lower compared to the standard condition with entity mentions, $p < 0.0001$. when looking at figure 13, we can observe that we find a larger amount of overfitting on all splits compared to our first dataset. Our loss over time is also way lower and can be seen in the full picture, which can be found in figure 16 in the appendix. Compared to our dataset with 10 entities, we do see that our dataset with 5 entities without entity mentions can achieve higher accuracy.

Table 16: Final training and validation loss, accuracy and F1 score after 20 epochs given per split for CNN model using 10 entity dataset that is under the *without entity mentions* condition.

	Split 1		Split 2		Split 3	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.081	**0.727**	0.094	1.090	0.080	0.893
Accuracy	0.995	**0.707**	0.996	0.586	0.993	0.661
F1	0.995	**0.704**	0.997	0.560	0.992	0.620

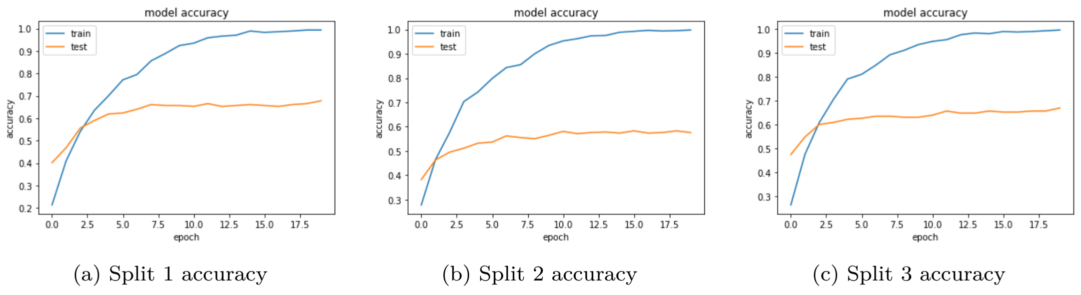
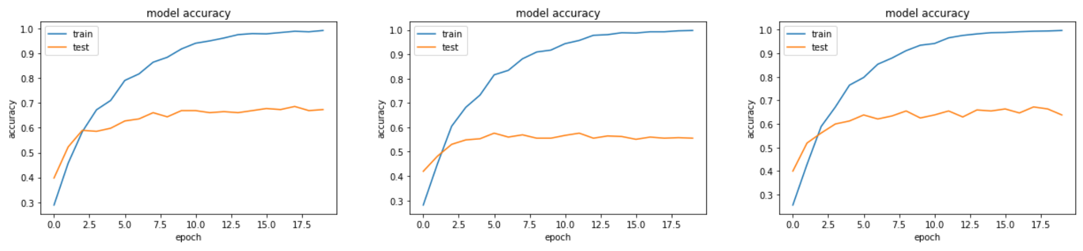


Figure 13: Training and Validation accuracy for CNN model using dataset with 5 entities under the *without entity mentions* condition. Results are shown per split over the duration of 20 epochs.

When looking at our third dataset where entity mentions are replaced, table 17, we see that the results found are similar to the dataset without entity mentions. Nonetheless, we do find a significant difference in the third split compared to the dataset with entity mentions. The dataset with replaced entity mentions performs better on the dataset without entity mentions, $p = 0.04$. Even so, when compared to the first dataset under the standard condition we see that this dataset performs significantly worse. The graphs for our third dataset in figure 29, show us that this dataset also has a lot of overfitting.

Table 17: Final training and validation loss, accuracy and F1 score after 20 epochs given per split for CNN model using 10 entity dataset that is under the *replaced entity mentions* condition.

	Split 1		Split 2		Split 3	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.086	**0.747**	0.090	1.010	0.081	0.792
Accuracy	0.995	0.715	0.999	0.593	0.997	**0.718**
F1	0.993	**0.709**	0.998	0.578	0.997	0.687



(a) Split 1 accuracy (b) Split 2 accuracy (c) Split 3 accuracy

Figure 14: Training and Validation accuracy for CNN model using dataset with 5 entities under the *replaced names* condition. Results are shown per split over the duration of 20 epochs.

4.3.3 Gibbs Sampling Dirichlet Multinomial Mixture Model

In table 18 we can see how our GSDMM model performs on our standard dataset using only 5 entities. The results show us that our model can classify 5 entities 10% better than when using our dataset with 10 entities. The highest accuracy we can achieve using 5 entities is **93,3%**. The highest accuracy achieved by our GSDMM is found using split 1, which was also the highest one when using the 10 entities dataset.

Table 18: Accuracy, precision, recall, F1 score and parameter settings given per split for GSDMM using 5 entities dataset that is under *standard* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Split 1	Split 2	Split 3
Accuracy	**0.933**	0.890	0.910
Precision	**0.933**	0.898	0.919
Recall	**0.933**	0.890	0.910
F1	**0.933**	0.887	0.907
α	0.3	1	0.1
β	0.8	0.6	0.5

When we remove the entity mentions, we can see that our results decrease significantly for all our splits (p <0.001) compared to the dataset with entity mentions. The highest accuracy found in table 19 is only **60,7%**. Similar accuracy can be found for the other 2 splits but slightly lower. All the splits, however, do show increased results compared to our results for our dataset with 10 entities.

Table 19: Accuracy, precision, recall, F1 score and parameter settings given per split for GSDMM using 5 entities dataset that is under the *without name* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Split 1	Split 2	Split 3
Accuracy	**0.607**	0.584	0.596
Precision	0.614	**0.630**	0.608
Recall	**0.607**	0.584	0.596
F1	0.557	**0.591**	0.584
α	0.7	1.0	0.6
β	0.8	0.7	0.6

Table 20 shows us the results for our model trained on the dataset where the entity mentions are replaced. The results found in our third dataset are very similar to the results in our dataset without entity mentions. The highest accuracy we can find for third dataset is **57,2%**, this accuracy is significantly lower than our dataset with entity mentions ($p < 0.001$).

Table 20: Accuracy, precision, recall, F1 score and parameter settings given per split for GSDMM using 5 entities dataset that is under the *replaced name* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Split 1	Split 2	Split 3
Accuracy	0.590	0.579	**0.592**
Precision	0.581	0.598	**0.622**
Recall	0.590	0.579	**0.592**
F1	0.516	0.569	**0.572**
α	0.3	0.7	0.5
β	0.8	0.6	0.7

4.4 Results for Muhammad dataset

The results for our Muhammad dataset uses multiple variations of the name Muhammad. Our models, when using the Muhammad dataset, search for one entity and try to classify it or it searches for two entities. The dataset exists of known Muhammads and unknown ones, and the known Muhammads are the ones that are being classified

4.4.1 Linear Regression Model

For our Muhammad dataset, we again can see that our best performing model is often the combined model, for that reason we have a broad overview of the combined model in table 21. In table 21 we can see that oftentimes our LR works just as well or even better when removing the entity mentions. However, to find out whether these results are any significant we have to take a more in-depth look into all our results found in the appendix.

Table 21: Precision, recall, F1 score and accuracy for Linear Regression model using Muhammad dataset for the standard, without and replaced conditions. Values are measured using the combination of the Distributed Memory model and the Distributed Bag of Words model.

	Prophet			Salah			Prophet & Salah		
	Stand.	With.	Repl.	Stand.	With.	Repl.	Stand.	With.	Repl.
Precision	0.762	**0.857**	0.782	0.778	**0.828**	0.722	0.715	**0.798**	0.713
Recall	0.762	**0.857**	0.779	0.778	**0.826**	0.716	0.705	**0.790**	0.685
F1	0.762	**0.857**	0.777	0.778	**0.825**	0.712	0.707	**0.789**	0.688
Accuracy	0.762	**0.857**	0.778	0.778	**0.825**	0.714	0.705	**0.789**	0.684

Table 39 shows us how our LR model performs when we use our Muhammad dataset. Using the dataset under standard condition, our LR model can classify the prophet Muhammad and Muhammad Salah similarly. The best performing algorithm for our LR model is the combination of algorithm, which has **76,2%** accuracy for searching the prophet Muhammad and **77,8%** accuracy when searching for Muhammad Salah. The difference found for searching either the prophet or Salah is insignificant, $p = 0.405$. We also see that the only significant difference between models is between the DBOW and the combination of model ($p = 0.01$), the DM and combination are insignificant (p =0.508). For the dataset where we look for both the prophet and salah, we see no significant differences between model algorithms.

For our dataset without entity mentions, we see that our accuracy increases for all our test sets. However, the increase in accuracy is not significant. Table 40 shows that the combination of models always performs significantly better than the DBOW model. Compared to the DM model, the combination of the models has no significant increase in performance on all of our entity detection tests. The highest accuracy found when searching for the prophet Muhammad is **85,7%**. When searching for Salah we have an accuracy of **82.5%**. Searching for the prophet and Salah had an accuracy of **78,9%**.

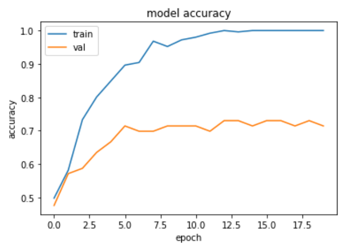
Looking at our dataset under the replaced entity mentions condition we see similar results as in the with entity mentions and without entity mentions dataset. Table 41 shows us that often time the combination works best, but not significantly better than the DM model. There are also no significant differences found between the replaced entity mentions dataset and the other two datasets.

4.4.2 Convolutional Neural Network Model

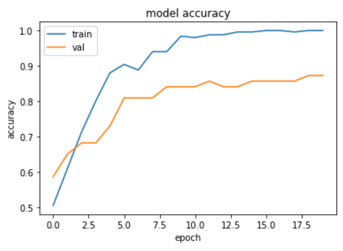
The results for our CNN model using the Mohammad dataset under standard condition show us that we cannot find any significant differences compared to our LR models. We see similar results for our CNN model and our LR model. Table 22 shows us an accuracy of **74%** is achievable when we look for both the prophet and Salah. To see how our accuracy grows over training epochs, we can take a look at figure 15. Figure 15 shows us that our accuracy for our validation set is overfitting more when looking only for the Prophet entity The graphs for the accuracy and loss in figure 30 show us that there is quite some overfitting happening when training our model.

Table 22: Final training and validation loss, accuracy and F1 score after 20 epochs given by dataset for CNN model using the Muhammad datasets that are under *standard* condition.

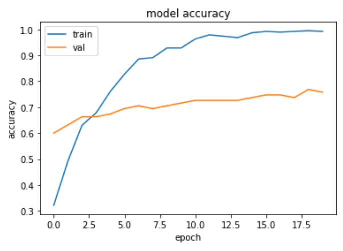
	Prophet		Salah		Prophet & Salah	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.061	**0.572**	0.048	0.604	0.106	0.615
Accuracy	0.996	0.714	0.988	0.698	0.992	**0.747**
F1	0.996	0.727	0.988	0.697	0.992	**0.740**



(a) Prophet accuracy



(b) Salah accuracy



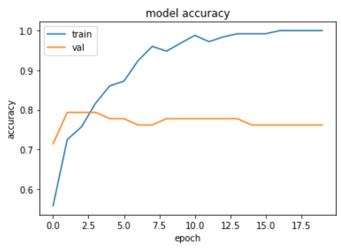
(c) Prophet & Salah accuracy

Figure 15: Training and Validation accuracy for CNN model using Muhammad dataset under *standard* condition. Results are shown per split over the duration of 20 epochs.

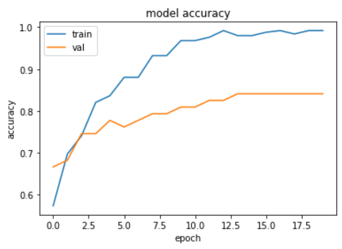
When looking at table 23, we see the results of our model using the dataset without entity mentions. Our model using the dataset without entity mentions seems to have better results for identifying Salah. Nonetheless, the results achieved without entity mentions are not significant compared to the results from our dataset with entity mentions ($p = 0.063$). Our dataset without entity mentions shows lower loss when training and validating, the results can be seen in figure 31

Table 23: Final training and validation loss, accuracy and F1 score after 20 epochs given by dataset for CNN model using the Muhammad datasets that are under the *without entity mentions* condition.

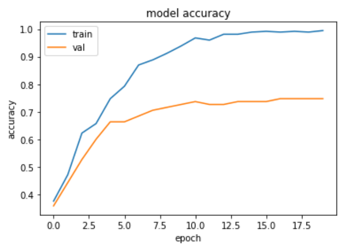
	Prophet		Salah		Prophet & Salah	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.044	0.585	0.055	**0.351**	0.097	0.571
Accuracy	1.000	0.762	0.992	**0.841**	0.995	0.747
F1	1.000	0.773	0.992	**0.848**	0.993	0.741



(a) Prophet accuracy



(b) Salah accuracy



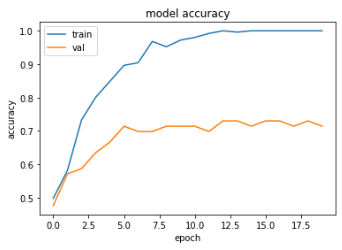
(c) Prophet & Salah accuracy

Figure 16: Training and Validation accuracy for CNN model using Muhammad dataset under *without entity mentions* condition. Results are shown per split over the duration of 20 epochs.

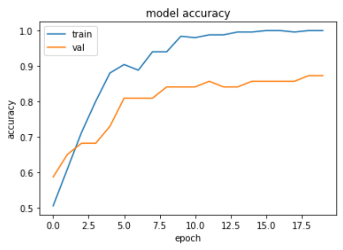
Our dataset with replaced entity mentions, however, does have a significant increase in accuracy compared to our dataset with entity mentions when looking to identify Salah. Table 24 shows us that our model can achieve **84,8%** accuracy when looking for Salah ($p = 0.043$). Other than an increase in finding Salah, we do not see any significant increases by replacing the entity mentions in our dataset. Our model loss and accuracy increase for our model over time behave similarly to our model when removing the entity mentions. We see slightly less overfitting for our model compared to our standard condition. Figure 32 shows us how the model loss and accuracy change over time.

Table 24: Final training and validation loss, accuracy and F1 score after 20 epochs given dataset for CNN model using the Muhammad datasets that are under the *replaced entity mentions* condition.

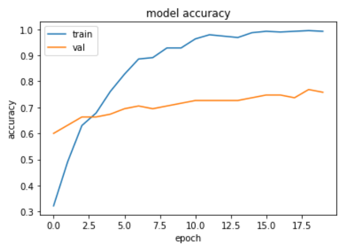
	Prophet		Salah		Prophet & Salah	
	Train	Val.	Train	Val.	Train	Val.
Loss	0.042	0.537	0.041	**0.319**	0.084	0.607
Accuracy	1.000	0.714	1.000	**0.873**	0.992	0.758
F1	1.000	0.682	1.000	**0.879**	0.994	0.757



(a) Prophet accuracy



(b) Salah accuracy



(c) Prophet & Salah accuracy

Figure 17: Training and Validation accuracy for CNN model using Muhammad dataset under *without entity mentions* condition. Results are shown per split over the duration of 20 epochs.

4.4.3 Gibbs Sampling Dirichlet Multinomial Mixture Model

In our GSDMM model we see that under normal conditions, all our datasets have similar results. Table 25 shows us that the highest accuracy is **79.4%** and it is acquired when trying to identify Salah.

Table 25: Accuracy, precision, recall, F1 score and parameter settings given per dataset for GSDMM using Muhammad datasets that are under *standard* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Prophet	Salah	Prophet & Salah
Accuracy	0.778	**0.794**	0.768
Precision	**0.826**	0.802	0.798
Recall	0.778	**0.794**	0.768
F1	0.770	**0.793**	0.762
α	0.7	0.6	0.8
β	0.7	0.1	0.7

When we remove the entity mentions, we see that our dataset looking for the prophet only increases significantly ($p < 0.001$). Table 26 shows that the accuracy when training our model to look for the prophet is the highest amongst the entities and is set at **89,9%** compared to the **77,8%** found in the standard dataset.

Table 26: Accuracy, precision, recall, F1 score and parameter settings given per dataset for GSDMM using Muhammad datasets that are under the *without entity mentions* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Prophet	Salah	Prophet & Salah
Accuracy	**0.889**	0.825	0.674
Precision	**0.889**	0.833	0.701
Recall	**0.889**	0.825	0.674
F1	**0.889**	0.824	0.655
α	0.1	0.3	0.6
β	0.8	0.1	1.0

Replacing the entity mentions shows a significant decrease in our model when searching for prophet Muhammad, compared to removing the entity mentions ($p = 0.04$). We also have a significant decrease when looking for both entity mentions compared to the standard dataset,$p = 0.02$. Although we see a very high increase in our model results when looking for Salah, the difference seen here is not significant ($p > 0.05$)

Table 27: Accuracy, precision, recall, F1 score and parameter settings given per dataset for GSDMM using Muhammad datasets that are under the *replaced entity mentions* condition. α and β given are those that performed best out of tested parameters ranging from 0 to 1 with interval 0.1

	Prophet	Salah	Prophet & Salah
Accuracy	0.794	**0.905**	0.737
Precision	0.834	**0.912**	0.800
Recall	0.794	**0.905**	0.737
F1	0.787	**0.904**	0.708
α	0.7	0.5	0.8
β	1.0	0.5	0.8

4.5 Overview of model results

To summarize our results, we were able to see that our models behave differently based on the number of entities needed to classify. Looking at table 28 we see that when we try to distinguish 10 entities under standard conditions, our CNN model is the best performer no matter what split is used (p <0.005). When lowering our amount of entities to 5, we can see that our CNN model is still the best performer. However, the TM model performs much better on 5 entities than on 10 entities. The performance of the CNN model and the TM model start resembling each other, when the number of entities is lowered. The only significant difference between the TM model and the CNN model is on the third split task, $p = 0.011$. With an even lower amount of entities, such as in the Muhammad task, we see that the TM model performance increases. The differences between the models on the Muhammad task are often not significant but there is one significant difference found on the Muhammad task when trying to classify both entities. The biggest difference found is between the LR and TM models, where we can see a 6% increase in accuracy when using the TM model ($p = 0.05$).

Table 28: The accuracy of all models performing tasks under the standard condition. Significant results are marked with an asterisk.

		LR	**CNN**	**TM**
10 Entities	**split 1**	0.609	**0.943***	0.774
	split 2	0.401	**0.890***	0.800
	split 3	0.438	**0.883***	0.801
5 Entities	**split 1**	0.607	**0.962**	0.933
	split 2	0.563	**0.931**	0.890
	split 3	0.641	**0.955***	0.910
Muhammad	**Prophet**	0.762	0.714	**0.778**
	Salah	0.778	0.698	**0.794**
	Prophet & Salah	0.705	0.747	**0.768**

The increased performance of our models using the standard conditions when lowering the number of entities is expected. Having a lower amount of entities to cluster or identify, simplifies the complexity of the task for all models. For the 10 and 5 entity datasets we can also see that, when searching for variations only, our models do not perform as well as using a standard 80% - 20% split (split 1). Nonetheless, the results found for our splits show that the models can find variations using the main entity mentions only. To get a good comparison of how similar or how different our models behave compared to each other and the baseline results, we can look at figure 18. In the histograms found in figure 18 we can see that our linear model performs way worse on the 10 and 5 entity dataset, but starts to perform almost as well as the other models when we use the Muhammad dataset. We can also see that our models always significantly outperform the baseline models when available.

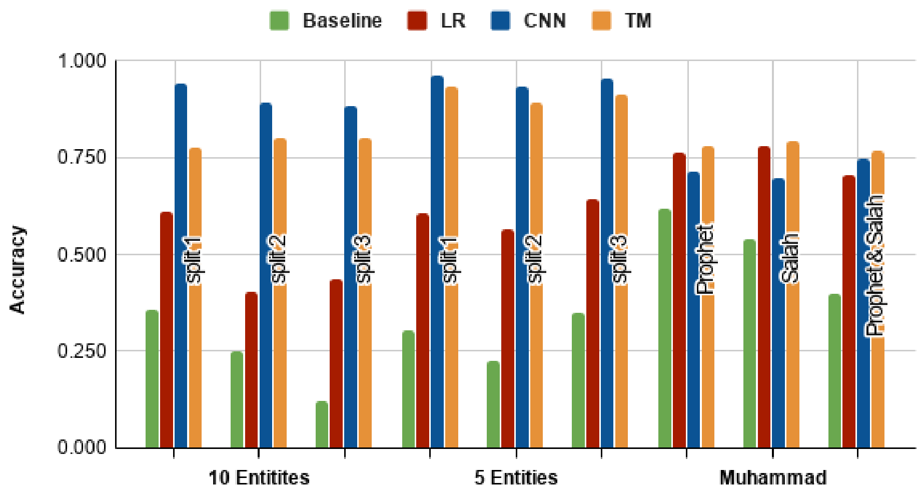


Figure 18: Accuracy for all models for each split using all datasets under the *standard* condition compared with Bag-of-words baseline model

Analyzing our dataset without entity mentions, found in table 29, shows us the same trend of results as seen in our standard dataset. We again see that CNN is the best performing model on the 5 and 10 entities tasks and that the TM model is the best performer on the Muhammad task. Nonetheless, we still see that our models perform significantly worse on the datasets which satisfy the without entity mentions condition than on the datasets which satisfy the standard condition. Compared to the results from the standard dataset, the model results found when using the datasets which satisfy the without entity mentions condition indicates that model performance is significantly lower when dealing with 5 or 10 entities ($p < 0.001$). The results of our models using the without entity mention dataset can be seen in figure 29. When looking at our Muhammad classification task, where we only try to identify 1 or 2 entities, we see that our model results are not significantly different from our standard condition. In some cases, such as the TM model when identifying either the Prophet or Salah entity using data that satisfies the without entity mentions condition, we can see some non-significant increase in accuracy when compared with the standard condition($p > 0.05$).

The results we see in our datasets without entity mentions is also an expected result, having no entity mentions in the dataset does increase the complexity as the context from different entities can be similar. The increased complexity in differentiating between entities explains why our 10 and 5 entities task has way worse performance. However, when we have different entities with the same name and different contexts we can see that our models do not perform significantly worse without entity mentions. Our Muhammad dataset has even shown increases in classification when removing entity mentions albeit non-significant ones. Compared to our baseline models, which did not score significantly better or worse for any condition, we see that all our models can significantly outperform them. Our results showed us that our models did not perform similarly as the key-word matching model used for the Muhammad dataset. We used the Bag-of-words model as our comparison baseline model, seeing as it can be applied to all our tasks and gives us a good comparison with the other tasks. Figure 19 shows us how our models compare to each other and the baseline Bag-of-words model. From the data gathered in our results section and the graphs

Table 29: The accuracy of all models performing tasks under the *without entity mentions condition*. Significant results are marked with an asterisk.

		LR	CNN	TM
	split 1	0.488	**0.571**	0.509
10 Entities	**split 2**	0.364	**0.502**	0.454
	split 3	0.391	**0.533**	0.508
	split 1	0.586	**0.707***	0.607
5 Entities	**split 2**	0.448	**0.586**	0.584
	split 3	0.571	**0.661***	0.596
	Prophet	0.857	0.762	**0.889**
Muhammad	**Salah**	0.825	**0.841**	0.825
	Prophet & Salah	**0.798**	0.747	0.674

in our current section, we can clearly see that the Bag-of-words baseline model is clearly under performing compared to our models. The baseline key-word matching model seems to be somewhat closer to our models but still falls flat when trying to distinguish both the entities.

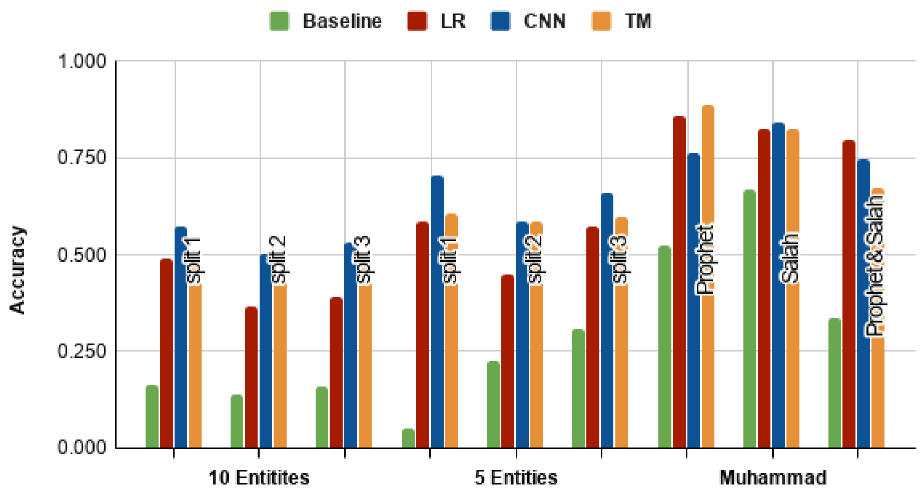


Figure 19: Accuracy for all models for each split using all datasets under the *without entity mentions* condition compared with Bag-of-words baseline model

Our datasets when under the replaced entity mentions condition show that they have no significant differences compared to the dataset without entity mentions. We see very similar trends in the histograms found in figure 19 and figure 20. Having all entity mentions replaced was also intended to be used as control group, giving us the ability to see if our models do not show any strange behaviors when we use a very simple entity mention for all our entities. The results in table 30 show that again CNN is the best performer on the 10 and 5 entities tasks, but also has good performance on classifying both the entities in the Muhammad task. The good performance on the Muhammad task by our CNN network, however, is not significant compared to the performance of our TM model($p = 0.701$).

Table 30: The accuracy of all models performing tasks under the *replaced entity mentions condition.* Significant results are marked with an asterisk.

		LR	CNN	TM
	split 1	0.450	**0.590**	0.472
10 Entities	**split 2**	0.311	**0.509**	0.478
	split 3	0.385	**0.549**	0.479
	split 1	0.628	**0.715***	0.590
5 Entities	**split 2**	0.446	**0.593**	0.579
	split 3	0.555	**0.718***	0.592
	Prophet	0.778	0.714	**0.794**
Muhammad	**Salah**	0.762	0.873	**0.905**
	Prophet & Salah	0.684	**0.758**	0.737

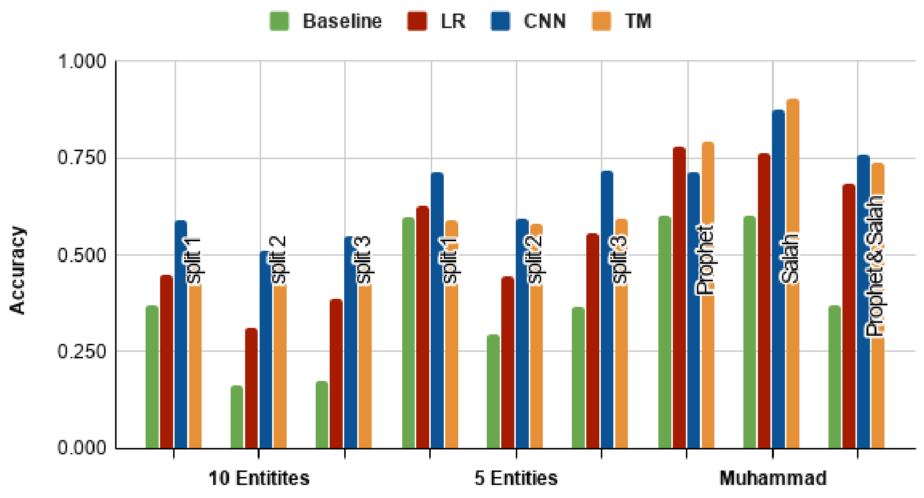
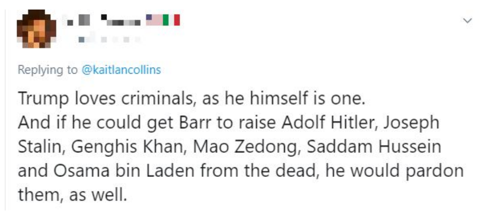


Figure 20: Accuracy for all models for each split using all datasets under the *replaced entity mentions* condition compared with Bag-of-words baseline model

Seeing as our CNN has shown to be the best performer on most tasks, we have also taken a look at the classification of each entity separately. In table 31 we see the F1 score for our CNN model based on each entity under the standard and without entity mentions condition. A quick look at the table shows us that many entities are classified way worse when the entity mentions are omitted. In split 1 with the standard condition, we see that all entity mentions are being classified with very high F1 scores. In the without entity mentions condition, we see that the F1 score is much lower for all the entity mentions. The biggest drop is found for the entity "Mahmoud Ahmadinejad". However, the entity "Recep Tayyip Erdogan" still performs at a reasonable rate when the entity mention is omitted. In the second split, in which our models try to find new variations for entity mentions, we see that in the standard condition our model performs worse than in split 1. Entities such as "Mahmoud Ahmadinejad" and "Yitzhak Rabin" have way lower F1 scores compared to their scores in the first split. Removing the entity mentions has the same effect as in the first split. Split 3 has a similar performance as split 2 but has slightly higher scores on average. To get an even better indication, of which entities might be getting confused, we can look at figure 22. In the confusion matrix, we can see that the entity "Mahmoud Ahmadinejad" is being predicted more often as the entities "Osama Bin Laden", "Recep Tayyip Erdogan" and "Saddam Hussein". We can also see that the model seems to confuse the entities "Saddam Hussein" and "Osama Bin Laden" more often when the entity mention is removed. The confusion of entities found might be somewhat confusing at first glance. For example, Saddam Hussein and Osama Bin Laden are both terrorists but they operated within different countries using different regimes. It should, therefore, be easy to distinguish them from context. However, when we take a look at some example tweets in which the entities are found, we realize why some confusion is happening. Figure 21 shows us an example of a tweet in our dataset in which multiple entities are mentioned. It is not unreasonable to believe that more of such tweets can be found within our dataset. In our dataset, the tweet we gave as an example would be tagged as both the Saddam Hussein entity and the Osama Bin laden entity. There are 24 more examples of tweets containing multiple entities. Although the number of tweets containing multiple entities are small, they could still influence our models. Having identical contextual information for some entities can be very confusing for the model and can explain why some entity mentions are being confused more often when only given context. There is also another reason why the confusion matrix shows the wrong classification of entities, we describe some of the issues more in-depth later on in our conclusion. The results seen in our confusion matrix reveal that some entities might have too similar contexts to classify correctly, meaning that more information besides context is needed to get a perfect classification.

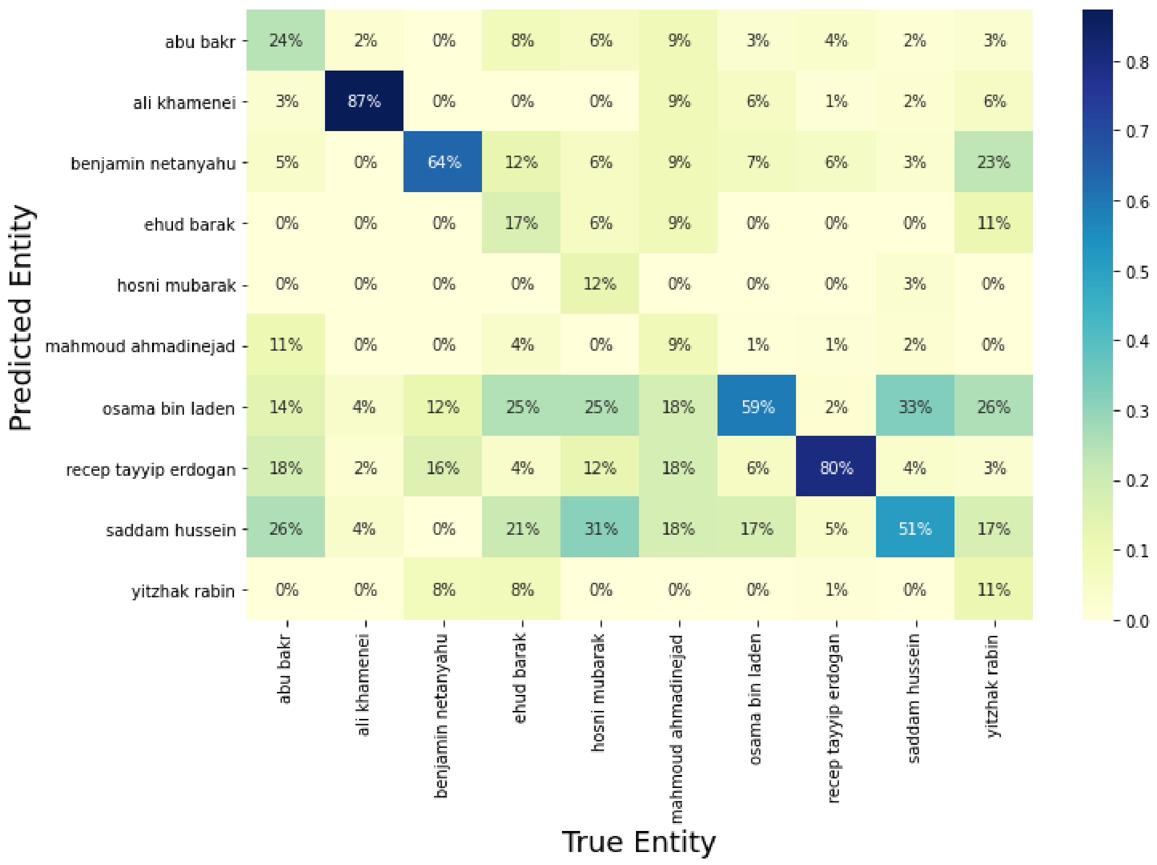

Replying to @kaitlancollins

Trump loves criminals, as he himself is one.
And if he could get Barr to raise Adolf Hitler, Joseph Stalin, Genghis Khan, Mao Zedong, Saddam Hussein and Osama bin Laden from the dead, he would pardon them, as well.

Figure 21: Example tweet showing multiple entities in one tweet.

Table 31: F1 scores for CNN model given for each entity. Scores are given for each split under the *standard* and *without entity mentions* conditions

	Split 1		**Split 2**		**Split 3**	
	Standard	**Without**	**Standard**	**Without**	**Standard**	**Without**
abu bakr	0.961	0.692	0.800	0.322	0.844	0.370
ali khamenei	1.000	0.729	0.946	0.727	0.963	0.825
benjamin netanyahu	0.914	0.580	0.800	0.356	0.813	0.423
ehud barak	0.913	0.564	0.667	0.240	0.900	0.588
hosni mubarak	0.950	0.303	0.769	0.333	0.714	0.222
mahmoud ahmadinejad	0.944	0.286	0.250	0.048	0.000	0.000
osama bin laden	0.938	0.460	0.939	0.531	0.939	0.504
recep tayyip erdogan	0.978	0.800	0.959	0.767	0.932	0.819
saddam hussein	0.908	0.471	0.922	0.461	0.882	0.432
yitzhak rabin	0.929	0.593	0.500	0.200	0.667	0.167

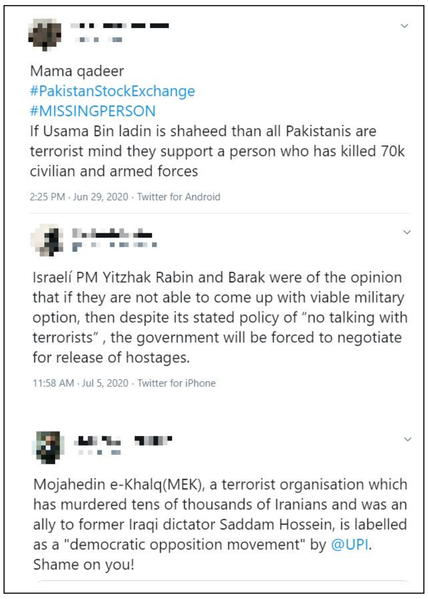


	abu bakr	ali khamenei	benjamin netanyahu	ehud barak	hosni mubarak	mahmoud ahmadinejad	osama bin laden	recep tayyip erdogan	saddam hussein	yitzhak rabin
abu bakr	24%	2%	0%	8%	6%	9%	3%	4%	2%	3%
ali khamenei	3%	87%	0%	0%	0%	9%	6%	1%	2%	6%
benjamin netanyahu	5%	0%	64%	12%	6%	9%	7%	6%	3%	23%
ehud barak	0%	0%	0%	17%	6%	9%	0%	0%	0%	11%
hosni mubarak	0%	0%	0%	0%	12%	0%	0%	0%	3%	0%
mahmoud ahmadinejad	11%	0%	0%	4%	0%	9%	1%	1%	2%	0%
osama bin laden	14%	4%	12%	25%	25%	18%	59%	2%	33%	26%
recep tayyip erdogan	18%	2%	16%	4%	12%	18%	6%	80%	4%	3%
saddam hussein	26%	4%	0%	21%	31%	18%	17%	5%	51%	17%
yitzhak rabin	0%	0%	8%	8%	0%	0%	0%	1%	0%	11%

Figure 22: Confusion matrix for split 2 under the *without entity mentions* condition, which shows the percentage of predicted entities for each true entity. Numbers are the percentages of true mentions.

5 Conclusion and Discussion

Classifying entity mentions in short texts, such as tweets, can be a challenging task. Having no comparable previous research, made evaluating the results of our models troublesome. Because of the lack of comparable research, we created baseline models to compare our results to those of existing techniques and methods. We saw that our baseline Levenshtein model could get reasonable results when looking for entity mention variations, but this could only be achieved if it would have been combined with a perfect named entity recognizer. However, perfect named entity recognition has not been fully achieved for short texts and there have been no indications that this will happen soon. Another problem that we encountered was testing our datasets using the without entity mention or replaced entity mention condition. We did not test our Levenshtein model on our tasks without- or replaced entity mentions as the results would not have shown anything of interest as either no matches would have been found or the same distance would have consistently been found. For our Muhammad task, for example, we already mentioned that a Levenshtein model would perform badly because it can not classify tweets using context. Performing our tests using Levenshtein turns classification into a syntactic task, however, a semantic approach is needed as context similarity plays an important role in the detection of entities.



[redacted]

Mama qadeer
#PakistanStockExchange
#MISSINGPERSON
If Usama Bin ladin is shaheed than all Pakistanis are terrorist mind they support a person who has killed 70k civilian and armed forces

2:25 PM · Jun 29, 2020 · Twitter for Android

[redacted]

Israelí PM Yitzhak Rabin and Barak were of the opinion that if they are not able to come up with viable military option, then despite its stated policy of "no talking with terrorists" , the government will be forced to negotiate for release of hostages.

11:58 AM · Jul 5, 2020 · Twitter for iPhone

[redacted]

Mojahedin e-Khalq(MEK), a terrorist organisation which has murdered tens of thousands of Iranians and was an ally to former Iraqi dictator Saddam Hossein, is labelled as a "democratic opposition movement" by @UPI. Shame on you!

Figure 23: Examples of tweets with similar contexts

5.1 Model performance

To answer how well our proposed models have performed, we first take a look at how the number of entities searched for can impact our models under the standard conditions. Looking at 5 instead of 10 entities does not significantly impact the performance of our CNN model. We see that our accuracy in all of our splits is very similar to our CNN model. For our LR model, we see that our model scores better on our second and third split when reducing the entities. Our TM does show significant improvement when reducing the number of entities from 10 to 5. Our Muhammad task, in which we want to look for 2 entities with the same mentions, shows us that our models all do not perform better than on the 5 or 10 entity tasks. However, even though the number of entities found here is lower, we cannot conclude that a very low amount of entities performs way worse. In our Muhammad task, we look at entities with the same mentions. But, in our 10 and 5 entities task, all the entities have very different entity mentions that are distinguishable.

Looking at our tasks, performed under non-standard conditions, we see that the number of entities might influence model performance. Both the conditions, in which we remove entity mentions and replace entity mentions, show an improvement in accuracy when we lower the number of entities. The improvement in accuracy suggests that when we look at classifying short texts including entity mentions, our model needs the entity mentions to be in the texts as they carry crucial information for classification. The results found are not unexpected, seeing as we already mentioned that our entities would likely have a similar context. In the three example tweets in figure 23, we can see how the word terrorist is used in the context of three different entities. It is not unlikely that more entities have similar words found around them, making our models believe that they are the same entity due to their context similarity. To strengthen the argument of context similarity mattering for our models, we look at our models' performances on our Muhammad task. We see that when we try to classify entities with the same mentions but completely different contexts, our models perform similarly no matter what condition is used. From our result, we can see that our models use both context and entity mentions. However, when we have 10 entities to classify, the similarity of their context can disrupt the models' performances. By not removing entity mentions we can use both the context and entity mention to classify tweets, allowing us to get acceptable results when identifying entity mentions without being affected by the number of entities. From our analysis, we conclude that when using entities with similar contexts without entity mentions, the number of entities do matter. We are also able to conclude that our CNN model is the best performing model in general and has no issues classifying up to 10 entities.

5.2 Entity mention variations

We have also observed how well our models can detect new entity mentions. Previous research has focused on detecting entities and their mentions by using corpora such as Wikipedia. Our results, however, for the standard condition in table 28 show us that when trained on only the main entity instead of all entities, our CNN can still accurately classify the mentions of entities. The differences found between splits do not seem to be significant. The results from our third split, however, do suggest that training on some of the variants helps to detect new ones more accurately. Seeing no differences between splits is a good indicator that our models can detect new variations based on known variations. When removing or replacing the entity mentions we see the same pattern as mentioned above for the standard condition. Our first split is used as a control measure, seeing as all entity mentions are used in there, and is the best performer. However, the difference found between splits is very small for our CNN model under all conditions. The results found show us that our CNN model acts robustly because not many entity variations are needed to get good results. The CNN model seems to efficiently use both the entity mention and the context of the entity mention, allowing the model to detect new variations. Detecting new variations efficiently can be a very useful tool for entity linking tasks.

To get an even better indication of how context influences the performance of our models, we can review the results found for our Muhammad task in tables 28, 29, 30. In our standard condition, found in table 28, we see that the best performing model is the TM model. The other models had no significant difference in the solo entity detection, but when looking for both entities we see that there is a significant difference between the LR and TM models. The task here was to find our entities in a dataset with other unknown entities, our models showed that they could all perform the task for detecting both entities with around 75% accuracy. Compared to our baseline models, all of our models perform significantly better. Our keyword matching model is the best baseline performer, which only shows a 51% accuracy for detecting both entities. For the single entity detection, our TM model is under most conditions the best performer and achieves significantly higher accuracy than our baseline keyword matching model. Our model results not only show us that they are better than the baseline models, but also that they seem to be using the context of entities to improve results. When we consider all our conditions, we see that our models do not have significant differences between conditions. In some cases, we can even see improvement in our accuracy. Our research indicates that context is very important for our models, even enabling us to differentiate between entities that carry four similar entity mentions.

Table 31 showed us how our CNN model behaved for each entity separately, which gave us insight into how contextual similarity might reduce entity variation detection. We can see that some entities are still problematic to classify even if the entity mentions are found within the tweets. For split 2 we saw that our entity "Mahmoud Ahmadinejad" was very hard to classify even though others were not. A possible explanation for the worse classification of "Mahmoud Ahmadinejad" could be due to the low amount of variations. For the other entities with low amounts of variations such as "Yitzhak Rabin and "Ehud Barak", we see similar occurrences but with less severity. Another reason for why "Mahmoud Ahmadinejad" might be classified worse than the other entities, could be due to its entity variations being very different from the main entity mention. Having very different entity variations could lead to them only occurring in non-similar contexts. Combined with the small number of variations, it could mean that when the entity variations are found in very different contexts the models simply fail to classify all the variation mentions completely and thus decreasing the accuracy heavily.

Our research results have given us insights into the role of many subjects that can play a part in classifying entity mentions within short texts. The most important information we have gained is that context can be used by different model architectures to identify entities within a tweet. We have also seen that by using context, we can find different entity mentions because of their context similarity. Being able to detect different entity mentions can be a very helpful tool when trying to get all possible information on an entity. We also noticed that when we want to train and search on a larger number of entities, we need to include the entity mentions during training. If entity mentions are not included during training with a larger number of entities, the performance of the models will decline. Finally, we are also able to report that our models can differentiate between entities that have the same entity mentions.

5.3 Limitations and Future Work

5.3.1 Generalizability

Although our research has given us valuable insight into the field of entity classification and linking, we did, however, encounter some limitations. One issue was linked to the generalizability of our research. For our research, we created our datasets ourselves, which we had to gather using the Twitter API. The data we had gathered were all from the same period, meaning that our results might differ if we had collected data over multiple years. Context on entities might drastically change with time and it would have been very useful to know how our models would have performed if this was taken into consideration.

Not only is the period of gathering our data troublesome for the generalizability of our research, but the fact that we only tested our model on entity names from Semitic origin also limits the generalizability of our model. Nonetheless, we do believe that testing our models on Semitic entity mentions will not be significantly different from testing our models on Non-Semitic entity mentions, but we are unable to conclude this with complete certainty. Future research is needed to completely ensure that our model can be generalized for more types of data and entity origins. Additionally, we only used tweets as our short texts. Although tweets are a good source for short texts, they might not be a perfect representation of all types of short texts. Tweets are targeted towards an audience of followers to whom you might want to talk to differently than perhaps a friend. The writing style of a short text can depend on the audience on the receiving end. There are different types of short texts that all need to be considered to have completely generalizable research.

5.3.2 Dataset

Another issue that we encountered in our research was that of the amount of varying data. Due to the limitations of time and data, we were not able to collect as much different entity mentions as we had hoped for. We would have also liked to have more entities and tweets per entities, giving us the ability to train on a big dataset. By training on a smaller dataset and showing decent results on it, we do believe that our model shows promise for practical uses. In practical cases, users of our model would likely need to search for a small number of entities instead of a large number. However, it is important to be sure our model does not behave very differently if our corpus is larger.

To find solutions to our aforementioned limitations we propose the use of a larger and more detailed corpus. If given the time the corpus can be created by the user, but gathering existing tweets and filtering these might be a better alternative. Improving our corpus can be done in many ways. One way to create a better corpus is to have it contain entities with similar amounts of varieties. The range of varieties we have for our entities can be quite different for the currently used entities. Our lowest number of variations available is 2 and the highest is 28. It would increase the validity of our results if we had a more equal distribution of entity variations in our dataset. Increasing the validity of our results will lead to better models and potentially improve results of entity linkage and recognition tasks in the future.

5.3.3 Language

A subject we have not yet tackled is the subject of language. For our research we have used names originating from Semitic languages, we did not incorporate any names coming from other regions of the world. Sino-Tibetan languages are found in many East-Asian countries and have the second-largest amount of native speakers. Most Sino-Tibetan languages are very tonal, a characteristic that is not present in most Indo-European languages. Looking at transliterated texts from other language families, such as Sino-Tibetan, could show other interesting findings. Another language-related subject that we have yet to discuss is the use of written language. In our research, we used the Latin script as written language and we thus look at variations that existed because of transliteration. It would be interesting to perform a similar type of research on languages using another script, such as Arabic or Hebrew.

5.3.4 Models and features

Furthermore, we have also seen some limitations in our model architectures. CNN models have a vast array of hyper-parameters that can be tuned and adjusted to the designers' liking. For our research, we used an architecture that is commonly used for classification with slight adjustments which we believed are more suited for our tasks. Nonetheless, different options could have been considered if there was more theoretical information on suitable architectures. For example, the number of layers we decided on was chosen based on trial and error testing and previous experiences. More detailed research, however, is needed to conclude how successful our model parameters have been. In follow-up research, it might be interesting to create different models and test these to find optimal performances. It could be that other model types are needed for different styles of texts, finding information on model architecture is very valuable and could even translate to other classification tasks. In addition to changing the parameters, we could also look more in-depth at how our models are affected by the scale of data. We already see some differences in our model results when we alter the number of entities to classify, but we have not seen how our models respond to large amounts of training and testing data. We can speculate that our TM model might be less and less successful the larger our number of entities becomes while our CNN stays unaffected, but future research could show us if this is true. We also will gain information on how fast our models will perform when our dataset increases in size, which can also give us more information on each models' practical relevance.

Classification tasks can be done using different tools and characteristics, one aspect of language which we have not discussed before is Part-of-Speech. Part-of-Speech tagging, also known as grammatical tagging, is the process of attributing the correct part of speech to each word. Derczynski et al. (2013) have shown great improvement on Part-of-Speech tagging on noisy and sparse data, they have achieved an 88.7% tagging accuracy which seems to be very decent and could potentially be improved. For our future work, we propose to incorporate information on Part-of-speech in our models. Feldman et al. (2009) shows that they can classify speech genres with the help of Part-of-Speech statistics such as their frequency. The researchers were even able to show that they could identify some new types of text genres, on which the models were not trained. G. Wang et al. (2015) has also shown that they can perform sentiment analysis using a random subspace method based on Parts-of-Speech information. From their results the researchers believe that their method can reduce the bias and variance of their task, giving them a reason to believe that it might be a useful feature for other text classification problems.

5.3.5 Real world data

Another point of interest that we have not yet discussed is the number of variations that we have found in real data compared to all variations known. With the use of the Twitter API, we were able to find many variations for entities. Nonetheless, the database from which we have gathered the entities had way more available entity mentions. For example, the entity "Ali Khamenei" had 8 different entity mentions in our corpus. However, when we look in the JRC database, "Ali Khamenei" had 69 different variations. For other entities, we can see similar amounts that have not been found in tweets. We can thus see that we have only found a small portion of all the entity variations that are found in other databases. Nonetheless, knowing that not every variation of an entity is found can be valuable information for future researchers. Not needing to account for every possible variation will reduce workload and increase efficiency for many models. Although not every entity variation needs to be learned, there is a reason for us to believe that the dataset we created might be slightly biased. The context in which many different tweets are found seem to be slightly more educated, oftentimes entity mentions are found in tweets from news-outlets or accounts that discuss political themes. The bias found in our dataset has likely occurred due to searching on the full name entity mentions. It could be that different types of users mention entities by only using the first or last names. In our research, we did not look at partial entity mentions and therefore we cannot make any statements on this subject. A solution to create less bias could be to annotate and find datasets ourselves. Manually annotating datasets will allow us to find correct mentions of entities using only partial entity mentions. We will be able to validate if the tweets are mentioning the specified entity or an unknown entity. The proposed procedure is similar to how our data for the Muhammad dataset is gathered. Creating a dataset by manually having to check each tweet can be time-intensive and was, therefore, only used in a small subset of our research, but performing this on our main task with 10 entities could be interesting for future work.

5.3.6 Possible improvements

Aside from the limitations of our research, we have also found some interesting results which we believe can be improved. We see that our models can identify new variations very well when given entity mention, however, we have also seen that our baseline Levenshtein model could perform this just as well or even better sometimes. Seeing as our detection of new entity variations is not optimal we propose to improve our models to help find these more effectively. One idea to improve future models is by using more context information. By looking at specific types of words such as nouns in the context and then comparing them with other texts, we could find similarities in the grammatical structures of the texts and maybe improve our models. Using word types would need the use of a good working Part-of-Speech tagger. Entity classification using our proposed models combined with the research that was done by G. Wang et al. (2015), could be interesting in the future. The information based on Parts-of-Speech might help our models classify entity mentions better, seeing as they have also improved other tasks of classification.

Another possible improvement we could use for our models is to incorporate spelling and similarity checks for our context. We use our context to classify entities but we did not implement any features for our models to first look at how similar our words might be. Although embedding tools do give similar vectors for similar words, they might not be represented fully if trained by using our corpus or our pre-trained embedding vectors only. Using word to word semantic relatedness, Tsatsaronis et al. (2010) show with the help of a thesaurus, that their method outperforms lexicon-based methods when researching text relatedness. It might be interesting for our research to look at the ideas and options for semantic relatedness to help our models classify tweets more accurately, as their context can be compared more precisely.

Finally, it might also be beneficial to create a control group using humans. We have only compared our models with each other, but we do not have an indication of how humans would classify our tweets. It might be interesting to compare our results with human classification as this is ultimately what we want to achieve. Comparing results can be done by creating a smaller set and letting participants classify our tweets based on a list of entities. To get the best results, multiple participants should be used, to account for biases and errors.

5.4 Practical relevance

Finding entity mentions and variations can be very interesting and could have many practical applications. Being able to retrieve information on how entities are being mentioned or perceived by users on micro-blogging websites can be of value to companies, financial traders, or even governments. Companies often need to know how potential clients are perceiving their business. By being able to classify mentions of their business correctly, they could create sentiment analysis on all tweets in which they are mentioned and not only on those correctly spelled. Traders could benefit the same way as companies do, however, they would prefer a fast working model. Seeing as large amounts of money can be lost or gained in seconds and all potential information is important for decision making. Government sectors such as law enforcement could use tools that can detect entity mentions to gather information on entities that are mentioned differently. It is not unlikely that entities are mentioned using coded language in messaging. There might be a lot of confiscated data, which can be tedious to check manually. Having a tool that can correctly classify texts mentioning an entity no matter what variation, can be very useful. Even entities carrying the same mentions can be discerned, giving us reason to believe that our research results could be relevant. However, when looking for practical uses of any of these models, one must keep in consideration the time needed for the models. We see that although our TM model shows good results, it is significantly slower than the other models when the number of entities is increased. The LR and CNN models both have similar speeds, but CNN is way more accurate in many cases. The findings of our current research can potentially create a starting ground for more advanced forms of entity classifications and perhaps even recognition.

A Appendix

A.1 Entity mentions

Table 32: Entities with mention variations for the 10 entities dataset, percentages indicate the amount of variants found based on main name, followed by the absolute values, the Levenshtein distance compared to the main entity mention and the corresponding entity variation

main entity mention	Variations	
abu bakr	(83.8%, 155, 0, 'abu bakr')	(5.4%, 10, 1, 'abu bakar')
	(5.4%, 10, 1, 'abou bakr')	(5.4%, 10, 1, 'abu baker')
ali khamenei	(72.1%, 147, 0, 'ali khamenei')	(5.9%, 10, 7, 'seyyed ali khamenei')
	(5.9%, 10, 4, 'imam khamenei')	(5.9%, 10, 7, 'sayyed ali khamenei')
	(5.9%, 10, 6, 'seyed ali khamenei')	(2.9%, 6, 1, 'ali khamanei')
	(2.9%, 6, 1, 'ali khameni')	(2.5%, 5, 1, 'ali khamenai')
benjamin netanyahu	(71.0%, 120, 0, 'benjamin netanyahu')	(5.2%, 10, 6, 'bibi netanyahu')
	(5.2%, 10, 1, 'benjaminnetanyahu')	(5.2%, 10, 2, 'binyamin netanyahu')
	(4.1%, 7, 1, 'benyamin netanyahu')	(4.1%, 7, 1, 'benjamin natanyahu')
	(3.0%, 5, 1, 'benjamin nethanyahu')	
ehud barak	(91.5%, 107, 0, 'ehud barak')	(8.5%, 10, 1, 'ehud barack')
hosni mubarak	(93.6%, 88, 0, 'hosni mubarak')	(6.4%, 6, 1, 'husni mubarak')
mahmoud ahmadinejad	(85.4%, 82, 0, 'mahmoud ahmadinejad')	(9.4%, 9, 7, 'mahmud ahmad')
	(5.2%, 5, 9, 'ahmadi nejad')	
osama bin laden	(55.6%, 179, 0, 'osama bin laden')	(3.11%, 10, 1, 'osama bin ladan')
	(3.11%, 10, 2, 'usama bin ladin')	(3.11%, 10, 1, 'osam bin laden')
	(3.11%, 10, 1, 'usama bin laden')	(3.11%, 10, 1, 'osama bin ladin')
	(3.11%, 10, 1, 'osama bin ladens')	(3.11%, 10, 8, 'ben ladden')
	(3.11%, 10, 7, 'ben laden')	(3.11%, 10, 6, 'bin laden')
	(3.11%, 10, 7, 'bin ladin')	(3.11%, 10, 1, 'osama ben laden')
	(3.11%, 10, 1, 'osama bin-laden')	(2.8%, 9, 1, 'ossama bin laden')
	(2.5%, 8, 3, 'oussama ben laden')	(1.9%, 6, 3, 'usame bin ladin')
recep tayyip erdogan	(54.3%, 120, 0, 'recep tayyip erdogan')	(4.5%, 10, 7, 'tayip erdogan')
	(4.5%, 10, 6, 'tayyip erdogan')	(4.5%, 10, 8, 'recep tayyip')
	(4.5%, 10, 7, 'tayyiperdogan')	(4.5%, 10, 7, 'tayyib erdogan')
	(4.5%, 10, 9, 'recep tayip')	(4.5%, 10, 1, 'recep tayyib erdogan')
	(4.1%, 9, 8, 'tayib erdogan')	(4.1%, 9, 1, 'recep tayip erdogan')
	(3.6%, 8, 7, 'recep erdogan')	(2.3%, 5, 8, 'tayep erdogan')
saddam hussein	(58.9%, 178, 0, 'saddam hussein')	(3.3%, 10, 1, 'sadam hussein')
	(3.3%, 10, 1, 'saddam hussain')	(3.3%, 10, 1, 'saddam husseins')
	(3.3%, 10, 1, 'saddamhussein')	(3.3%, 10, 1, 'saddam husein')
	(3.3%, 10, 2, 'saddam hussien')	(3.3%, 10, 2, 'saddam hossain')
	(3.3%, 10, 2, 'sadam husein')	(3.3%, 10, 2, 'saddam husain')
	(3.0%, 9, 1, 'saddam hussen')	(2.7%, 8, 2, 'sadamm hussein')
	(2.0%, 6, 1, 'saddam hossein')	(2.0%, 6, 1, 'saddam houssein')
	(1.7%, 5, 2, 'saddam hoessein')	
yitzhak rabin	(93.8%, 135, 0, 'yitzhak rabin')	(6.2%, 9, 1, 'itzhak rabin')

A.2 10 entities

A.2.1 LR model

Table 33: Precision, recall, F1 score and accuracy for Linear Regression model using 10 entities under *standard* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Split 1			Split 2			Split 3		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	**0.592**	0.395	0.588	**0.353**	0.278	0.347	0.236	0.304	**0.332**
Recall	0.585	0.402	**0.590**	0.416	0.329	**0.421**	0.229	0.340	**0.341**
F1	**0.587**	0.393	0.586	**0.354**	0.252	0.339	0.225	0.305	**0.319**
Accuracy	0.585	0.439	**0.609**	**0.401**	0.311	0.393	0.315	0.388	**0.438**

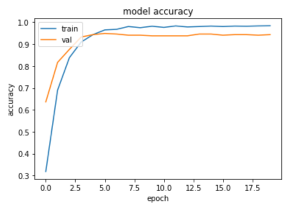
Table 34: Precision, recall, F1 score and accuracy for Linear Regression model using 10 entities under the *without entity mentions* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Split 1			Split 2			Split 3		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.338	0.412	**0.460**	0.261	0.286	**0.306**	0.249	0.279	**0.311**
Recall	0.321	0.403	**0.473**	0.298	0.309	**0.362**	0.268	**0.309**	0.305
F1	0.328	0.403	**0.464**	0.250	0.257	**0.295**	0.251	0.273	**0.299**
Accuracy	0.342	0.429	**0.488**	0.314	0.327	**0.364**	0.334	0.341	**0.391**

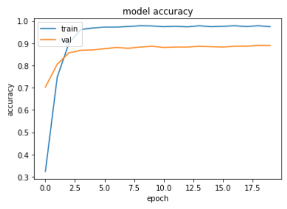
Table 35: Precision, recall, F1 score and accuracy for Linear Regression model using 10 entities under the *replaced entity mentions* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Split 1			Split 2			Split 3		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.354	0.409	**0.444**	0.193	0.262	**0.273**	0.235	0.283	**0.309**
Recall	0.333	0.394	**0.442**	0.207	0.307	**0.311**	0.217	0.329	**0.337**
F1	0.340	0.397	**0.441**	0.175	0.242	**0.249**	0.219	0.287	**0.297**
Accuracy	0.340	0.415	**0.450**	0.230	0.307	**0.311**	0.290	0.363	**0.385**

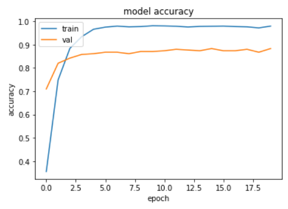
A.2.2 CNN model



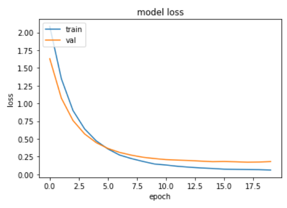
(a) Split 1 accuracy



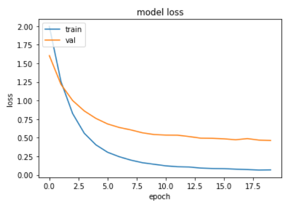
(b) Split 2 accuracy



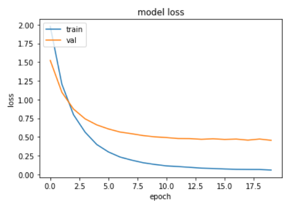
(c) Split 3 accuracy



(d) Split 1 loss

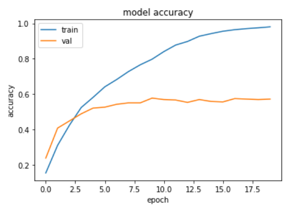


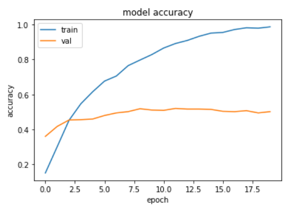
(e) Split 2 loss

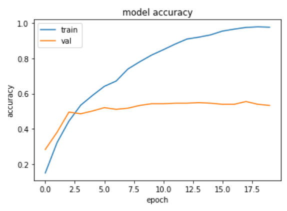


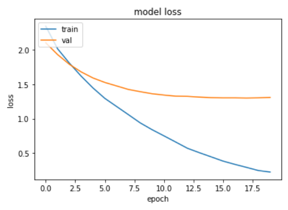
(f) Split 3 loss

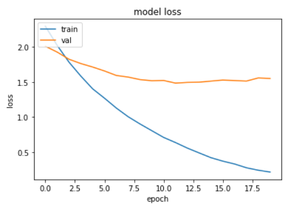
Figure 24: Training and Validation loss/accuracy for CNN model using dataset with 10 entities under standard condition. Results are shown per split over the duration of 20 epochs.


(a) Split 1 accuracy


(b) Split 2 accuracy


(c) Split 3 accuracy


(d) Split 1 loss


(e) Split 2 loss

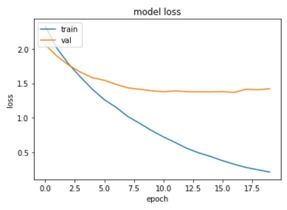
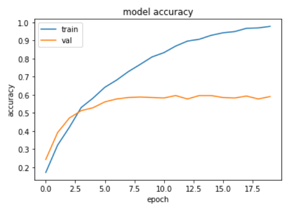
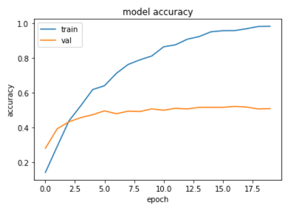
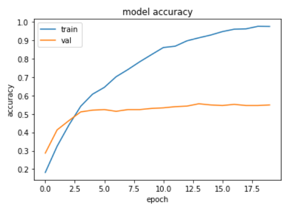
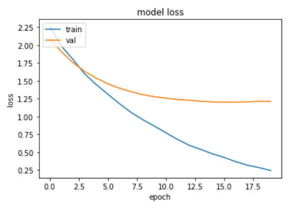

(f) Split 3 loss

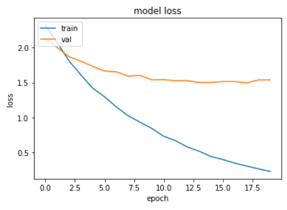
Figure 25: Training and Validation loss/accuracy for CNN model using dataset with 10 entities under the *without names* condition. Results are shown per split over the duration of 20 epochs .


(a) Split 1 accuracy


(b) Split 2 accuracy


(c) Split 3 accuracy


(d) Split 1 loss


(e) Split 2 loss

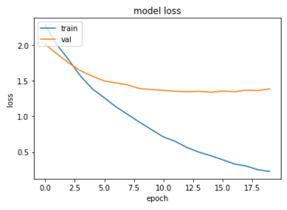

(f) Split 3 loss

Figure 26: Training and Validation loss/accuracy for CNN model using dataset with 10 entities under the *replaced names* condition. Results are shown per split over the duration of 20 epochs.

A.3 5 entities

A.3.1 LR model

Table 36: Precision, recall, F1 score and accuracy for Linear Regression model using 5 entities under *standard* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Split 1			Split 2			Split 3		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.421	**0.627**	0.594	**0.551**	0.442	0.542	0.556	0.560	**0.632**
Recall	0.409	**0.612**	0.583	0.554	0.483	**0.631**	0.552	0.611	**0.712**
F1	0.411	**0.614**	0.585	**0.535**	0.427	**0.535**	0.542	0.556	**0.637**
Accuracy	0.393	**0.607**	0.577	**0.563**	0.451	0.526	0.588	0.576	**0.641**

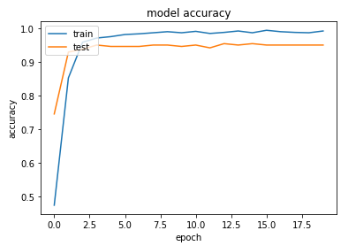
Table 37: Precision, recall, F1 score and accuracy for Linear Regression model using 5 entities under the *without entity mentions* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Split 1			Split 2			Split 3		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.428	0.581	**0.589**	0.415	**0.456**	0.444	0.390	0.538	**0.565**
Recall	0.416	**0.588**	0.583	0.434	**0.590**	0.509	0.388	0.591	**0.629**
F1	0.420	**0.580**	**0.580**	0.402	**0.437**	0.418	0.387	0.541	**0.570**
Accuracy	0.410	0.573	**0.586**	0.446	**0.448**	0.437	0.437	0.555	**0.571**

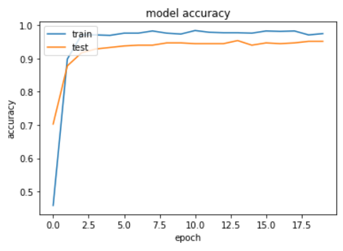
Table 38: Precision, recall, F1 score and accuracy for Linear Regression model using 5 entities under the *replaced entity mentions* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Split 1			Split 2			Split 3		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.484	0.509	**0.631**	0.305	0.429	**0.466**	0.310	0.435	**0.530**
Recall	0.482	0.522	**0.640**	0.400	0.491	**0.549**	0.309	0.495	**0.579**
F1	0.481	0.512	**0.628**	0.297	0.406	**0.446**	0.299	0.425	**0.531**
Accuracy	0.481	0.506	**0.628**	0.322	0.423	**0.467**	0.322	0.445	**0.555**

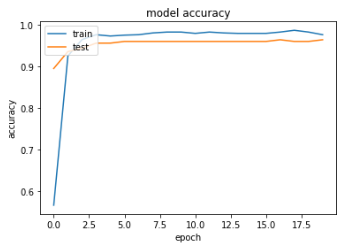
A.3.2 CNN model



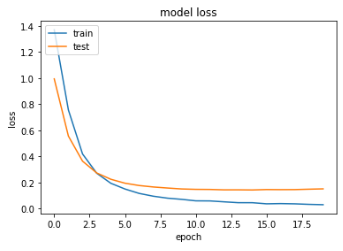
(a) Split 1 accuracy



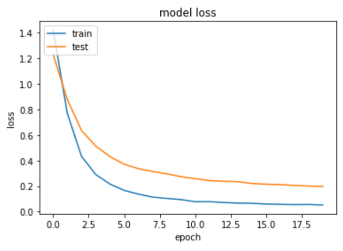
(b) Split 2 accuracy



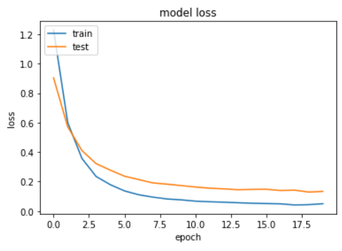
(c) Split 3 accuracy



(d) Split 1 loss

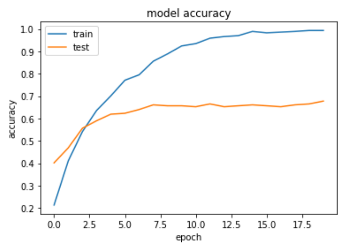


(e) Split 2 loss

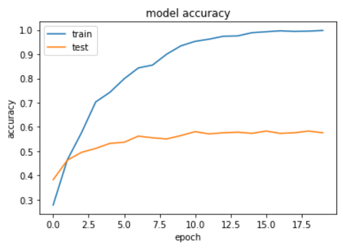


(f) Split 3 loss

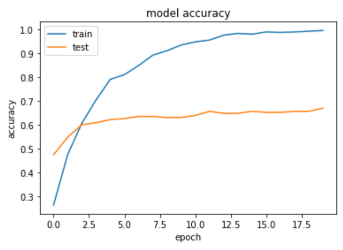
Figure 27: Training and Validation loss/accuracy for CNN model using dataset with 5 entities under standard condition. Results are shown per split over the duration of 20 epochs.



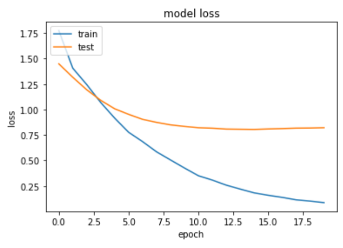
(a) Split 1 accuracy



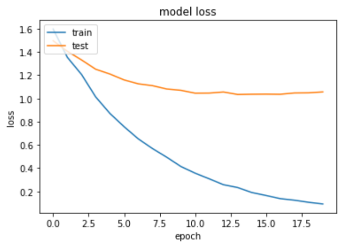
(b) Split 2 accuracy



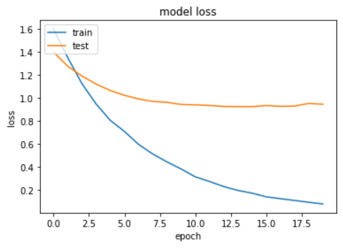
(c) Split 3 accuracy



(d) Split 1 loss

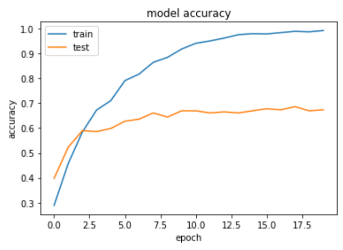


(e) Split 2 loss

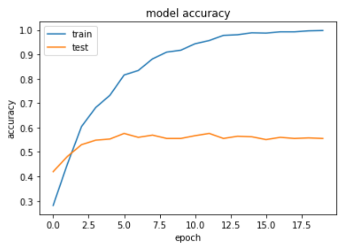


(f) Split 3 loss

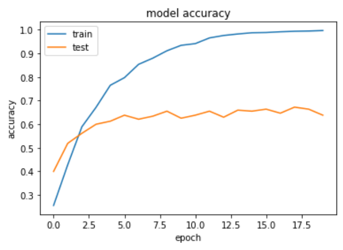
Figure 28: Training and Validation loss/accuracy for CNN model using dataset with 5 entities under the *without names* condition. Results are shown per split over the duration of 20 epochs.



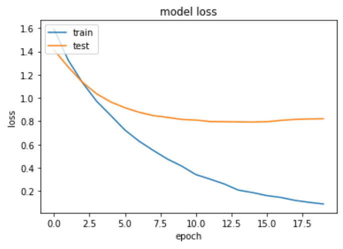
(a) Split 1 accuracy



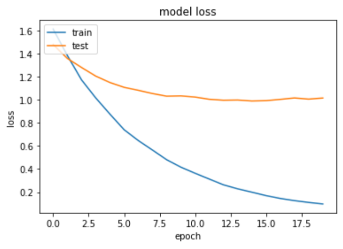
(b) Split 2 accuracy



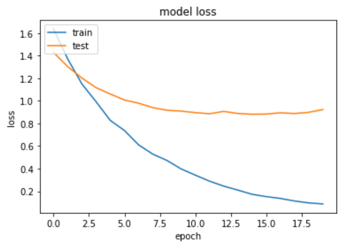
(c) Split 3 accuracy



(d) Split 1 loss



(e) Split 2 loss



(f) Split 3 loss

Figure 29: Training and Validation loss/accuracy for CNN model using dataset with 5 entities under the *replaced names* condition. Results are shown per split over the duration of 20 epochs.

A.4 Muhammad Dataset

A.4.1 LR model

Table 39: Precision, recall, F1 score and accuracy for Linear Regression model using the Muhammad datasets under *standard* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Prophet			Salah			Prophet & Salah		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.662	0.718	**0.762**	0.621	0.730	**0.778**	0.607	**0.722**	0.715
Recall	0.653	0.713	**0.762**	0.620	0.730	**0.778**	0.601	0.704	**0.705**
F1	0.646	0.712	**0.762**	0.618	0.730	**0.778**	0.597	0.704	**0.707**
Accuracy	0.651	0.714	**0.762**	0.619	0.730	**0.778**	0.600	**0.705**	**0.705**

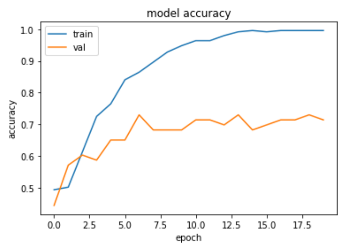
Table 40: Precision, recall, F1 score and accuracy for Linear Regression model using the Muhammad datasets under the *without entity mentions* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Prophet			Salah			Prophet & Salah		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.700	0.796	**0.857**	0.685	0.826	**0.828**	0.619	0.702	**0.798**
Recall	0.699	0.794	**0.857**	0.681	**0.826**	**0.826**	0.620	0.696	**0.790**
F1	0.698	0.793	**0.857**	0.681	**0.825**	**0.825**	0.619	0.693	**0.789**
Accuracy	0.698	0.794	**0.857**	0.683	**0.825**	**0.825**	0.621	0.695	**0.789**

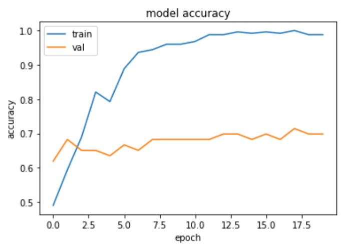
Table 41: Precision, recall, F1 score and accuracy for Linear Regression model using the Muhammad datasets under the *replaced entity mentions* condition. Values are measured for the Distributed Bag of Words model(DBOW), Distributed Memory model (DM) and a combination of both models.

	Prophet			Salah			Prophet & Salah		
	DBOW	DM	Comb.	DBOW	DM	Comb.	DBOW	DM	Comb.
Precision	0.747	0.736	**0.782**	0.492	**0.762**	0.722	0.531	0.693	**0.713**
Recall	0.745	0.731	**0.779**	0.492	**0.762**	0.716	0.536	**0.685**	**0.685**
F1	0.745	0.729	**0.777**	0.492	**0.762**	0.712	0.531	0.683	**0.688**
Accuracy	0.746	0.730	**0.778**	0.492	**0.762**	0.714	0.537	**0.684**	**0.684**

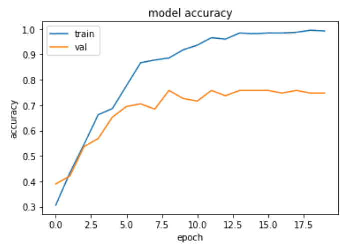
A.4.2 CNN model



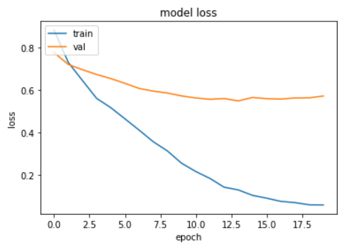
(a) Prophet accuracy



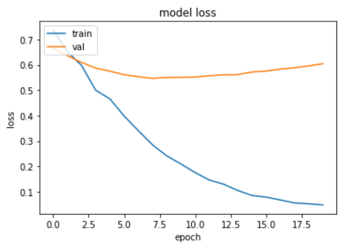
(b) Salah accuracy



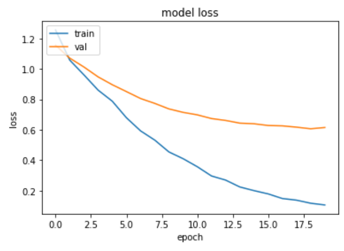
(c) Prophet & Salah accuracy



(d) Prophet loss

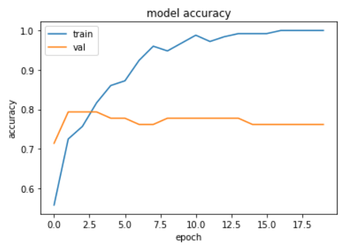


(e) Salah loss

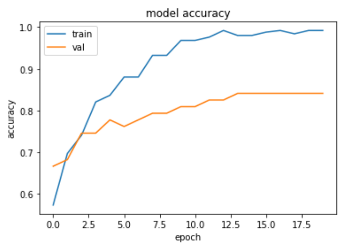


(f) Prophet & Salah 3 loss

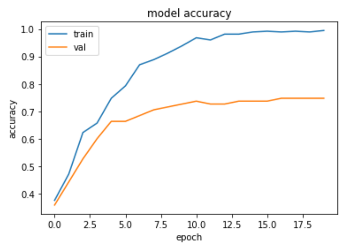
Figure 30: Training and Validation loss/accuracy for CNN model using "Muhammad" dataset under standard condition. Results are shown per entity or combination over the duration of 20 epochs.



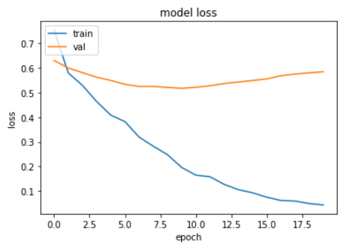
(a) Prophet accuracy



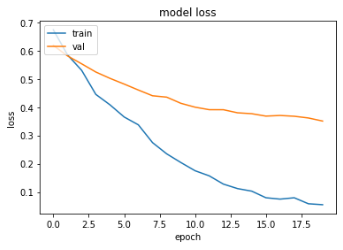
(b) Salah accuracy



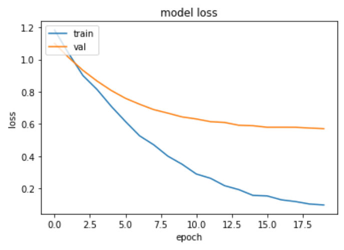
(c) Prophet & Salah accuracy



(d) Prophet loss

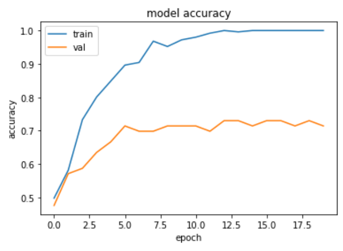


(e) Salah loss

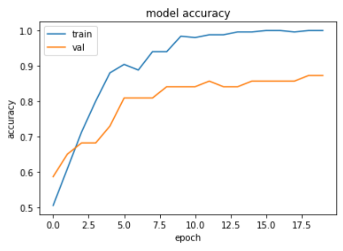


(f) Prophet & Salah 3 loss

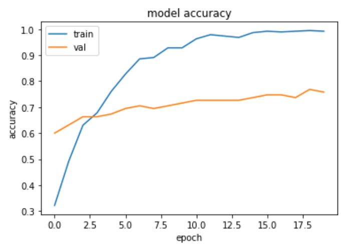
Figure 31: Training and Validation loss/accuracy for CNN model using "Muhammad" dataset under *without names* condition. Results are shown per entity or combination over the duration of 20 epochs.



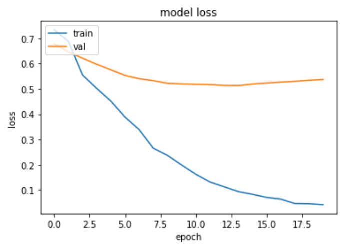
(a) Prophet accuracy



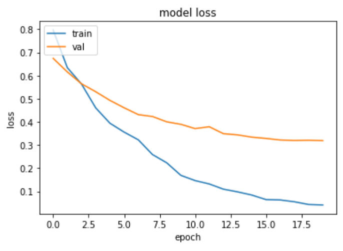
(b) Salah accuracy



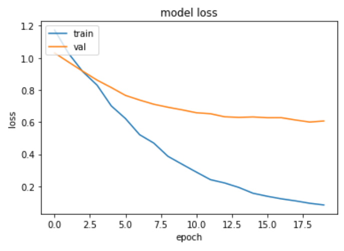
(c) Prophet & Salah accuracy



(d) Prophet loss



(e) Salah loss



(f) Prophet & Salah 3 loss

Figure 32: Training and Validation loss/accuracy for CNN model using "Muhammad" dataset under the *replaced names* condition. Results are shown per entity or combination over the duration of 20 epochs.

Bibliography

Abad, A., Ortega, A., Teixeira, A., Mateo, C. G., Hinarejos, C. D. M., Perdigão, F., Batista, F., & Mamede, N. (2016). *Advances in Speech and Language Technologies for Iberian Languages: Third International Conference, IberSPEECH 2016, Lisbon, Portugal, November 23-25, 2016, Proceedings* (Vol. 10077). Springer.

Asgari, E., & Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, *10*(11), e0141287.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

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

Celik, I., Abel, F., & Houben, G.-J. (2011). Learning semantic relationships between entities in Twitter, In *International conference on web engineering*. Springer.

Chakrabarti, T., Saha, S., & Sinha, D. (2013). DNA multiple sequence alignment by a hidden Markov model and fuzzy Levenshtein distance based genetic algorithm. *International Journal of Computer Applications*, *73*(16), 26–30.

Chaudhuri, S., & Agrawal, S. (2004). Generalized keyword matching for keyword based searching over relational databases [US Patent 6,792,414]. Google Patents.

Chen, M., Jin, X., & Shen, D. (2011). Short text classification improved by learning multi-granularity topics, In *Twenty-second international joint conference on artificial intelligence*. Citeseer.

Chowdhury, S. D., Bhattacharya, U., & Parui, S. K. (2013). Online handwriting recognition using Levenshtein distance metric, In *2013 12th international conference on document analysis and recognition*. IEEE.

Christen, P. (2012). The data matching process, In *Data matching*. Springer.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning, In *Proceedings of the 25th international conference on machine learning*.

Cucchiarelli, A., & Velardi, P. (2001). Unsupervised named entity recognition using syntactic and semantic contextual evidence. *Computational Linguistics*, *27*(1), 123–131.

Derczynski, L., Maynard, D., Rizzo, G., Van Erp, M., Gorrell, G., Troncy, R., Petrak, J., & Bontcheva, K. (2015). Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, *51*(2), 32–49.

Derczynski, L., Ritter, A., Clark, S., & Bontcheva, K. (2013). Twitter part-of-speech tagging for all: Overcoming sparse and noisy data, In *Proceedings of the international conference recent advances in natural language processing ranlp 2013*.

Elgibali, A. (2005). *Investigating Arabic: Current parameters in analysis and learning* (Vol. 42). Brill.

Feldman, S., Marin, M. A., Ostendorf, M., & Gupta, M. R. (2009). Part-of-speech histograms for genre classification of text, In *2009 ieee international conference on acoustics, speech and signal processing*. IEEE.

Ganguly, D., Roy, D., Mitra, M., & Jones, G. J. (2015). Word embedding based generalized language model for information retrieval, In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval*.

Hachey, B., Radford, W., & Curran, J. R. (2011). Graph-based named entity linking with Wikipedia, In *International conference on web information systems engineering*. Springer.

Hachey, B., Radford, W., Nothman, J., Honnibal, M., & Curran, J. R. (2013). Evaluating entity linking with Wikipedia. *Artificial intelligence*, *194*, 130–150.

Hensher, D. A., Rose, J. M., & Greene, W. H. (2012). Inferring attribute non-attendance from stated choice data: Implications for willingness to pay estimates and a warning for stated choice experiment design. *Transportation*, *39*(2), 235–245.

Jung, J. J. (2012). Online named entity recognition method for microtexts in social networking services: A case study of Twitter. *Expert Systems with Applications*, *39*(9), 8066–8070.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents, In *International conference on machine learning*.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals, In *Soviet physics doklady*.

Lhoussain, A. S., Hicham, G., & Abdellah, Y. (2015). Adaptating the Levenshtein distance to contextual spelling correction. *International Journal of Computer Science and Applications*, *12*(1), 127–133.

Limsopatham, N., & Collier, N. (2016). Bidirectional LSTM for named entity recognition in Twitter messages.

Maloney, J., & Niv, M. (1998). Tagarab: A fast, accurate Arabic name recognizer using high-precision morphological analysis, In *Computational approaches to semitic languages*.

Mani, K., Verma, I., Meisheri, H., & Dey, L. (2018). Multi-document summarization using distributed bag-of-words model, In *2018 ieee/wic/acm international conference on web intelligence (wi)*. IEEE.

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation, In *Empirical methods in natural language processing (emnlp)*. http://www.aclweb.org/anthology/D14-1162

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211–252.

Schraagen, M., & Kosters, W. (2014). Record linkage using graph consistency, In *International workshop on machine learning and data mining in pattern recognition*. Springer.

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013). Parsing with compositional vector grammars, In *Proceedings of the 51st annual meeting of the association for computational linguistics (volume 1: Long papers)*.

Steinberger, R., Pouliquen, B., Kabadjov, M., & Van der Goot, E. (2013). Jrc-names: A freely available, highly multilingual named entity resource. *arXiv preprint arXiv:1309.6162*.

Su, G.-y., Li, J.-h., Ma, Y.-h., & Li, S.-h. (2004). Improving the precision of the keyword-matching pornographic text filtering method using a hybrid model. *Journal of Zhejiang University-Science A*, *5*(9), 1106–1113.

Tolles, J., & Meurer, W. J. (2016). Logistic regression: Relating patient characteristics to outcomes. *Jama*, *316*(5), 533–534.

Tsatsaronis, G., Varlamis, I., & Vazirgiannis, M. (2010). Text relatedness based on a word thesaurus. *Journal of Artificial Intelligence Research*, *37*, 1–39.

Wang, G., Zhang, Z., Sun, J., Yang, S., & Larson, C. A. (2015). POS-RS: A random subspace method for sentiment classification based on part-of-speech analysis. *Information Processing & Management*, *51*(4), 458–479.

Wang, J., Liu, P., She, M. F., Nahavandi, S., & Kouzani, A. (2013). Bag-of-words representation for biomedical time series classification. *Biomedical Signal Processing and Control*, *8*(6), 634–644.

Wichmann, S., Holman, E. W., Bakker, D., & Brown, C. H. (2010). Evaluating linguistic distance measures. *Physica A: Statistical Mechanics and its Applications*, *389*(17), 3632–3639.

Yin, J., & Wang, J. (2014). A dirichlet multinomial mixture model-based approach for short text clustering, In *Proceedings of the 20th acm sigkdd international conference on knowledge discovery and data mining.*