UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

eXascale Infolab

GraphEDM: A Graph-Based Approach to Disambiguate Entities in Microposts

Master Thesis

Prathyusha Nerella

Supervisor

Prof. Dr. Philippe Cudré-Mauroux

Department of Informatics, University of Fribourg

Co-Supervisors

Akansha Bhardwaj

University of Fribourg

Paolo Rosso

University of Fribourg

5.11.2020

Abstract

The use of microblogging platforms such as Twitter has been growing rapidly. With about 500M tweets published per day, tweets are becoming a valuable source of information for several tasks such as event detection, sentiment analysis, or opinion mining, and are being leveraged by many prominent organisations.

However, one must first be able to correctly capture the semantic content of a tweet prior to leveraging it for any automated analysis. Automatically understanding tweets is extremely challenging, as the information they contain is limited and insufficient for algorithms that need a larger context. In this work, we propose an approach that extends the context of a micropost by leveraging graph-based algorithms to further disambiguate the entities present in it. Our approach, GraphEDM, is divided into two phases. First, we use unsupervised clustering approaches to regroup tweets in semantic neighborhoods using embedding approaches. Next, each ambiguous entity in a cluster is iteratively disambiguated by leveraging a graph-based algorithm. Our experimental results reveal that GraphEDM outperforms the state of the art in tweet entity disambiguation by up to 15.13% on several gold standard datasets.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Twitter[1] is one of the widely used microblogging services that allows users to post personal messages in real-time. These messages, commonly referred to as tweets, form an important source of instant information on any topic, including celebrity gossip, entertainment, news, etc. Tweets as a whole represent a huge amount of unstructured data, which is leveraged by news agencies for detecting important events, companies for detecting the launch of new products, for opinion mining, etc. [36]. Consequently, the development of specific techniques for analyzing tweets has attracted considerable attention in recent years [1]–[3], [46], [59].

One of the important requirements for tweet analysis is understanding a tweet semantically, which typically involves Entity Disambiguation techniques. Entity Disambiguation is a sub-field of Information Extraction (IE) in which extracted mentions from the text are mapped onto existing entities in a reference Knowledge Base (KB). Some of the widely used Knowledge Bases in this context are Wikidata[2], DBpedia[3], and YAGO4 [40]. In this work, we focus on Entity Disambiguation from tweets, with Wikidata as our reference Knowledge Base.

Table 1.1a shows a few examples of tweets with a number of mentions (in bold) that need to be disambiguated. Entity Disambiguation is challenging for tweets as their content is very short, and often noisy and ambiguous. Furthermore, as seen in Table 1.1b, each mention in a tweet typically can be mapped onto multiple candidate entities. Only one of these candidate entities is the correct Wikidata entity for each mention.

In more detail, two key challenges make it hard to disambiguate entities in tweets:

[1] `https://twitter.com/`
[2] `https://www.wikidata.org/`
[3] `http://www.dbpedia.org/`

Example Tweet
NYC police shoot bystanders: **New York City** police officers trying to subdue an agitated man Saturday night fir... http://t.co/ozdEZvy2g1
RT @**BBCNews**: Norwegian police say man arrested at youth camp shooting is linked to the bombing in #**Oslo**
Western envoys tout deal on core of U.N. **Syria** draft; **Russia** denies: **UNITED FNATIONS** (Reuters) - After weeks of... http://t.co/4qALyno3SL
US intelligence on **Syria** gas attack 'unconvincing', says **Russia** http://t.co/RXmiEFBDTd

(a) Tweets with mentions in bold

Mentions	**Candidate Entities**	**Matching Entity**
New York City	Q60, Q1200081, Q3339042, Q38754	Q60
BBCNews	Q787216, Q1160945, Q4834860	Q1160945
Oslo	Q585, Q1794768, Q7107003, Q19321913,	Q585
Russia	Q34266, Q42195226, Q159, Q7381938	Q159
Syria	Q3509007, Q207118, Q21441670, Q858	Q858
UNITED NATIONS	Q7888316, Q7888319, Q1065, Q7888314	Q1065
US	Q53552040, Q30, Q1456659, Q28775762	Q30

(b) Mentions with their corresponding candidate entities and the correct entity from the Wikidata

Table 1.1: (a) Tweets with mentions are highlighted in bold. Our goal is to annotate each mention in a tweet with the corresponding entity in the Knowledge Base. (b) Mentions in the tweets may be ambiguous, which means that each mention in a tweet may have multiple candidate entities in the Knowledge Base. Matching entities for mentions are shown.

- With a limit of 280 characters, tweets fall into the category of short texts or microposts. Due to this character limit, users often tweet concisely, giving very little information about the context. Though often straightforward for human readers to understand, this makes automatic entity disambiguation very challenging on tweets.
- Tweets are informal in their style. The use of emojis, abbreviations, user names, hashtags or slang is common, which makes their interpretation difficult. Proper capitalization or grammatical rules are also not guaranteed, although they are often leveraged by automated disambiguation approaches.

Overall, context is an essential feature for entity disambiguation, but is often implicit in tweets. In this work, we propose an Entity Disambiguation approach that extends each tweet's context and leverages a graph-based technique to disambiguate mentions more effectively. We make the following contributions:

1.2 Motivation

Named Entity Disambiguation has been a research area since a long time. Linking is done on the entities from formal text, Web Tables, short messages and many more. Many supervised, semi-supervised and unsupervised works have been done providing state of the art systems in this field.

Named Entity Disambiguation in tweets is a research area in which more work can be done to increase the performance of overall NEEL process. Correct disambiguation of an entity in a tweet makes a tweet more meaningful and understandable. Good disambiguation techniques are needed to obtain good performance in NEEL process and in-turn use this information in other applications.

1.3 Thesis objectives

The objective of this thesis is to propose an entity disambiguation model for tweets. The main contributions from the thesis are as follows:

- Understand a state-of-the-art entity disambiguation system on Web Tables.
- Modify the algorithm and implement it to be able to use it for disambiguation in Tweets.
- Perform context extension on tweets in a dataset using different unsupervised clustering techniques and compare performance of disambiguation with these different clustering methods.
- Apply different text vectorization techniques on tweet texts and compare the performance of disambiguation with these techniques.
- Acquire seven different gold standard datasets on tweet entity disambiguation and modify the entities in them to link to the chosen Knowledge Base.
- Evaluate the implemented entity disambiguation systems for tweets with all the datasets.

As part of this thesis, we explore different text clustering techniques and find the technique that best suites the datasets. We use the clustering techniques with varying features and verify the generated clusters manually to make sure clustering is performed promptly. We also perform evaluation on the chosen state-of-the-art baselines with the datasets and compare the performance of disambiguation with the proposed GraphEDM.

On three datasets, all the variants of the implemented system perform better than the baselines. On two other datasets, many variants of the implemented method performe better than baseline methods. On the last two datasets, we observe satisfying results on different variants of the implemented system.

1.4 Thesis outline

In Chapter 2, we explain the basics of Named Entity Disambiguation. This chapter introduces Name Entity Extraction and Linking, listing few previous works tackling the same problem. We also present different clustering and text vectorization techniques that are used as part of this thesis. Finally, we also decriibe features of Knowledge Bases and provide details on the Knowledge Base that is used in GraphEDM.

In Chapter 3, we present the proposed graph-based technique - GraphEDM. The chapter contains the architecture of the proposed method with the detailed explanation about each of the sub-modules involved in it.

Chapter 4 contains different experiments conducted as part of the thesis with results. This chapter also details about the datasets used in the empirical evaluation of this thesis and data preparation method used. We also tabulate, analyze and compare the performance of different variants of the proposed system.

In chapter 5, finally, we conclude our thesis, present some of its limitations, and discuss potential future work.

Chapter

2

Background

Before developing a tool for disambiguation, it is important to understand the concept of Named Entity Extraction and Linking (NEEL). This chapter presents some basic terminology about tweets. This chapter also explains NEEL in detail and present few algorithms from previous works on NEEL in tweets.

2.1 Terminology

To understand entity disambiguation better, we should clearly distinguish the terms that are used to describe the process. Below are few terms with definitions:

- Named entity: a real world object that represents a person, organization, location and so on. An entity is represented with a proper name. This object may or may not have an entity representing it in a Knowledge Base.
- Entity: An entity represents a real world entity present in a Knowledge Base [52].
- Mention: A word or word phrase occurred in a text document that has the potential to be mapped to an entity in the Knowledge Base. In other words, a named entity occurred in a text document that can be linked to an entity.
- Candidate Entity: For a mention, an Entity Linking algorithm considers one or more entities in the Knowledge Base as potential candidates. Linking algorithm will be applied on these chosen candidates to map the mention and the mapped entity will be one of the candidate entities or NIL depending on the result of the algorithm.

Twitter and Tweets

Twitter is an American micro blogging and social networking service on which users post and interact with messages known as "tweets" [56]. As of 2018, Twitter had more than 321 million monthly active users [34]. Twitter generates huge amounts of tweet data everyday. This data can be used for research in different areas to produce useful information. This information is in turn useful in many upstream application domains.

2.2 Natural Language Processing

Natural Language Processing (NLP) is the process of modifying human written data to get valuable information for computers to understand. Works on NLP has started around 1950s. There are many NLP techniques most of which use Machine Learning algorithms to process data. NLP helps to build applications to solve many real world problems. Implementation of NLP processes is difficult than expected. Human language is generally unstructured and often abstract or inaccurate. While humans can easily master a language, the ambiguity and imprecise characteristics of the natural languages are what makes NLP difficult for machines to implement [16]. The level of ambiguity in few sentences is high making them difficult to understand even for humans. Below are the examples of such sentences that are ambiguous [37], [38]

- "The professor said on Monday he would give an exam."
- "The burglar threatened the student with the knife."

In the first sentence, it is difficult to say whether the exam is on Monday or the announcement regarding the exam was made on Monday. In the second sentence, it is not clear whether the student or burglar has the knife. Human reasoning can be helpful in the second case, as we know that the burglar will be in possession of a knife and not the student. These examples clearly explain the challenge in implementing an NLP system. There are many state-of-the-art systems for NLP that does sentence segmentation, tokenization, stemming and named entity recognition. Applications like automated text summarization, extracting sentiments from a text, word stemming use Natural Language Processing. NLP techniques are also being developed to work on multiple languages besides English.

2.3 Named Entity Extraction and Linking

Named Entity Extraction and Linking (NEEL) has attracted a lot of researchers from the beginning of this decade [25], [35], [43], [45], [53]. Many related works have been done on formal text like web documents and also on informal, short text like tweets and news articles. As discussed in the previous section, the IE task NEEL includes two sub tasks namely entity Named Entity Extraction and Named Entity Disambiguation. In the next subsection we shall see some previous works on NEE and NED.

2.3.1 Named Entity Extraction

Identification of valid mentions in the given text document is Named Entity Extraction (NEE). NEE algorithms use Ontology that contains semantic information about the entities in the text. The coverage of the Ontology used should be higher for a better precision in the NEE task. This task is usually extended. After extracting the entities in the text, they will be classified into categories. Some of these categories include Person, Organization, and Location. These tasks are together called Named Entity Recognition (NER) and this is also referred as Named

Entity Recognition and Classification (NERC) in few works.

Many algorithms have been designed for NER and some of them are Stanford NER tagger [28], DBpedia Spotlight [31], Zemanta[1]. Figure 2.1 shows an example NER output with Stanford NER tagger. These algorithms are considered as the state-of-the-art and attain high performance for NER on text. The best system for English news articles entering MUC-7 scored an F-measure of 93.39% while human annotators scored 97.60% and 96.95% [18], [29].

Many attempts have also been made for NER in tweets. Ritter et al [47] was one of the first algorithms for NER in Tweets. The algorithm performs parts-of-speech tagging, parsing and trains a SVM classifier for predicting effectiveness of capital words in a tweet. It also trains a CRF model with different features from tweets and mentions that are helpful for NER task. Different entity types are considered as classes for LabeledLDA classification which uses Bag-Of-Words (BOW) for entity type and mention in the tweet. The tweets that are used in this work are published as dataset and the dataset is used as gold standard in later NER research.

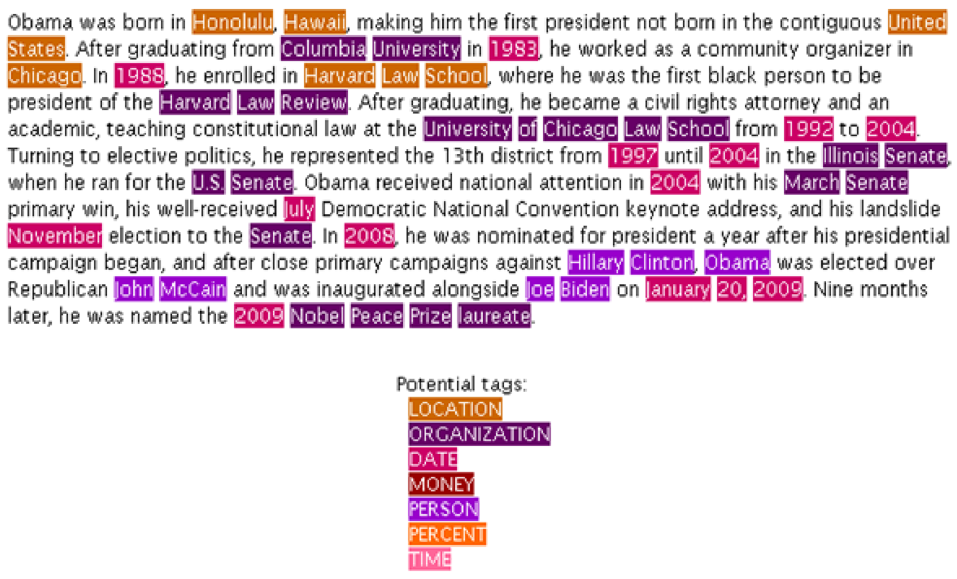
A graph based unsupervised algorithm, Twiner [24] uses global and local context for NER in tweets. The system uses Wikipedia and Web N-gram corpus for generating possible list of NEs in a tweet. The algorithm then assigns a rank to each candidate NE considering probability of actually being an entity. NEs with higher ranks have a higher chance of being an actual entity.

Contextual clustering of tweets is leveraged for Named Entity Extraction in **jung2012online**. This method uses "temporal associations" and "social associations" along with the tweet text similarity to guide the extraction. As the names suggest, temporal associations use the timestamp at which the tweets are posted, while social associations leverage the relationships between the Twitter users who post the tweets. The intuition behind this approach is that tweets published within the same time period and among the same groups share more similarities. In this thesis instead, the context of the tweets is extracted using multiple tweets by analyzing their textual content.

2.3.2 Named Entity Disambiguation

Named Entity Disambiguation (NED) involves mapping the mentions to an entity in a Knowledge Base. NED task have been studied separately for formal and informal texts as informal texts possess more challenges for disambiguation. Many tools that focus exclusively on Entity Disambiguation, use state-of-the-art Entity Extraction methods as the first step in their pipelines **hosseini2019implicit**, [11], [60]. Some of the recent works on NEEL focus on end to end process for entity extraction and linking [23].
These approaches have been applied on long documents like web pages and also on informal, shorter pieces of text like tweets and news articles. In the following, we start by discussing

[1]http://wandora.org/wiki/Zemanta_extractor



Obama was born in Honolulu, Hawaii, making him the first president not born in the contiguous United States. After graduating from Columbia University in 1983, he worked as a community organizer in Chicago. In 1988, he enrolled in Harvard Law School, where he was the first black person to be president of the Harvard Law Review. After graduating, he became a civil rights attorney and an academic, teaching constitutional law at the University of Chicago Law School from 1992 to 2004. Turning to elective politics, he represented the 13th district from 1997 until 2004 in the Illinois Senate, when he ran for the U.S. Senate. Obama received national attention in 2004 with his March Senate primary win, his well-received July Democratic National Convention keynote address, and his landslide November election to the Senate. In 2008, he was nominated for president a year after his presidential campaign began, and after close primary campaigns against Hillary Clinton, Obama was elected over Republican John McCain and was inaugurated alongside Joe Biden on January 20, 2009. Nine months later, he was named the 2009 Nobel Peace Prize laureate.

Potential tags:
LOCATION
ORGANIZATION
DATE
MONEY
PERSON
PERCENT
TIME

Figure 2.1: Stanford NER tagger [28] result for Wikipedia article of Obama

the related work on long documents and tables, before focusing on Entity Disambiguation approaches for short informal text and tweets.

2.3.2.1 Entity Linking on Documents and Tables

NED for formal text like Web Documents is well studied in research. The approaches use Wikipedia, DBpedia or YAGO knowledge bases for disambiguation. An entity annotation tool on Web tables is also discussed here.

AIDA

AIDA [60] is an online tool for entity disambiguation in text and tables through YAGO2 Knowledge Base [20]. The tool accepts text to disambiguate as input. AIDA uses Stanford NER [28] to identify the mentions in the text. Mentions can also be manually flagged in the text by putting them in double brackets.

AIDA performs disambiguation using a graph based approach. The graph is weighted, undirected and considers mentions and candidate entities as nodes. Mention-Entity edges have a similarity score which is a linear combination of two factors. The first one is prior probability of the entity which is collected using anchor texts and link targets in Wikipedia. The second factor is based on the overlap between contexts of mention and entity. Partial matches are also considered with a mechanism for penalizing them. The Entity-Entity edges have a weight which is the semantic relatedness between the entities. This is proportional to the number of incoming links that are shared between their Wikipedia articles.

After constructing the graph, greedy algorithm is used to reduce this graph to a dense one. In each iteration, teh method removes an entity node with the least weighted degree from the graph. The final entity node connected to the mention node is the disambiguated entity for that mention. The tool also provides run-time information and all intermediary results like similarity and coherence scores.

Annotating Web Tables

DoSeR [61] introduced a compelling global disambiguation approach for Web table entity disambiguation. The algorithm trains a Word2vec [33] model for all the entities and uses this model for calculating the similarity between the entities. A k-partite graph is constructed with all k candidate entities of mentions, with edges representing the similarity between the embeddings of the entities. Subsequently, PageRank [39] is applied to the graph to get a ranking of the nodes. The candidate entity with the highest score is considered as the disambiguated entity for the ambiguous mention under consideration. This approach was also used by [9] in their work on entity disambiguation in tables.

Eslahi et al. [10] optimized the above approach [61] by introducing an iterative model to perform the Web table annotation task. This algorithm uses Wikidata Knowledge Base for disambiguation. These algorithms consider the cosine similarity between the vector representations of the entities as the similarity between the entities. These vectors are called embeddings and they represent the semantic correlation between entities in the Knowledge Base. The algorithm for disambiguation has 2 steps: offline and online step.

In the offline step, the method genertaes a surface form pickle that maps all the entities in the Wikidata Knowledge Base with the surface forms or known names of the mentions. Thus the surface form pickle provides us with list of candidate entities for the given mention. Next, a Word2vec model is trained with all the candidate entities and their vector representations are stored. The training is done by extrapolating triplets from the Wikidata knowledge graph and considering each triplet as a sentence. The trained Word2vec model thus provides similarity between any two entities in the Knowledge Base.

In the online stage, the algorithm uses the Word2vec model and surface form pickles in the disambiguation process. Web Tables may contain multiple columns and only one column is considered as a label column. The annotation algorithm considers the values from the label column as mentions and annotates them with Knowledge Base entities. The iterative embedding algorithm constructs an initial graph with all the unambiguous candidate entities for the mentions in the table. Construction of a correct initial graph plays an important role in the correct disambiguation of the mentions with multiple entity candidates. In each iteration of the looping algorithm, candidates of one ambiguous mention are considered along with the initial graph. A k-partite graph is formed with k candidates for the mention considered. The graph contains no edges connecting the candidate entity nodes for the same mention. The weights on the edges are calculated using the model trained and PageRank [39] ranking algorithm is applied on the graph to get the ranking scores. The candidate entity with the highest score is considered as the disambiguated entity for the ambiguous mention in the loop. This entity node is then added to the initial graph at the end of the loop. The loop is repeated for all the ambiguous mentions in the label column of the table. The method uses type checking when candidate entities have the same ranking score and considers the candidate with most similar type in the graph is as the entity for the mention. The algorithm also uses other ranking algorithms along with PageRank to evaluate the method.

The iterative method also uses other columns which are called as Reference columns in the Web Table to elevate the context. The work also tries to minimize the complexity of the algorithm by using only one reference column at a time along with the label column in the table. The looping method using embeddings has shown better performance measures on the gold standard web table datasets when compared with the state-of-the-art baseline.

2.3.2.2 Entity Linking on Short, Informal Text

Named entity disambiguation is well studied in informal texts like tweets and news articles. Algorithms for formal text will not work well for short texts. The informal text is difficult to work with as it may not have more sense or meaning. Different supervised [25], [30], [32], semi supervised [21] and unsupervised [11], [12], [42] methods have been developed with different concepts for entity disambiguation in informal texts.

WAT-API
WAT-API is the successor of TageMe [12] a well-known entity disambiguation tool. The disambiguation process uses Wikipedia Knowledge Base and has 3 steps: Spotting, Disambiguation and Pruning.

In the spotting step, the annotator produces a list of mentions from the given text. A database of all the possible spots from Wikipedia is created beforehand in an offline process. The database also contains a list of candidate entities for every spot or mention present in the database. For each mention, the entity candidates are sorted according to the prior probability of the entity and mention. For each mention, the algorithm uses a diverse set of statistics to create this database. This data is then used to train a binary classifier − SVM with linear or RBF kernel. The classifier helps the spotter perform better.

For disambiguation step, WAT provides different disambiguation algorithms which are voting based and graph based. These algorithms uses context around the mention in the text provided as input. In the voting algorithm, all the candidate entities for each mention are assigned a score based on relatedness measure of the entities and prior probability of entity and the mention. WAT provides three variants in which either (a) trigram-similarity (jaccard) between the title of the entity and mention, or (b) the number of positive votes (number of times when the relatedness is greater than 0), or (c) both (a) & (b) indicators are used in addition to the semantic relatedness as a way to derive a score for the entity [42]. The graph based disambiguation algorithm uses a weighted undirected graph with mentions and candidate entities as nodes similar to AIDA as seen in the previous section. The weight of the edges between mention and its candidate entities is one of identity, commonness or context similarity values. The weight of the edges between entities is one of four different relatedness measures which included jaccard similarity, LSI vector cosine similarity. After constructing the graph, ranking algorithm is applied to get a highest weighted edge among all the edges between a mention and its candidate entities. WAT algorithm uses PageRank, Personalized Page Rank, HITS and SALSA algorithms for this purpose. This highest weighted edge connects the mention to its disambiguated entity.

WAT API tries to increase the precision of the algorithm by pruning the non-coherent annotations that are disambiguated in the previous step. The method calculates a synthetic metric which is the average of link probability of the mention and coherence of the annotation with the surrounding annotations. This metric of an annotation is compared against a set threshold value to decide whether to prune it or not. WAT also trains a classifier in this pruning for better performance.

WAT API also provides a mechanism for D2KB which performs disambiguation for the manually provided mentions along with text. The algorithm decides whether to perform disambiguation for all the mentions in the text or only for mentions passed as input and returns the disambiguated entities for the mentions. Unlike WAT-API, our approach, GraphEDM builds a graph with only entities (of mentions in a Knowledge Base) as nodes without considering the mention text directly. We show that our graph that uses only entities makes the disambiguation process better.

Entity linking of tweets based on dominant entity candidates
Entity linking of tweets based on dominant entity candidates (ELTDS) [11] is an unsupervised recent work on entity linking on tweets that has shows better performance on datasets. ELTDS works on the hypothesis that few and dominant entity candidates can be used for disambiguating the mentions in the tweets instead of all the available candidates. The algorithm performs 2 steps for annotation: an offline step and an online step.

In the offline step, the algorithm identifies the dominant entity candidates for the ambiguous mentions. Dominant candidates are the entity candidates for a mention that are frequently used within certain time period on twitter. The process for finding dominant entity candidates is divided into 2 steps again. The first step is term cluster detection and the second step is entity mapping. The algorithm adopts latent hypothesis that states that terms appearing in the same tweet have related semantics. First, a directed term dependency graph is formed with the terms from Twitter corpus. The algorithm uses all the n-grams from tweets as terms which are then considered as nodes of the graph. An edge between two nodes indicates that the terms have co-occurred in one or more tweets. The weight of an edge indicates conditional dependency between the terms considered. So, the pair of edges between same pair of nodes may have different weights. This graph provides a way to identify the terms occurring frequently with the mention. Considering a tweet with ambiguous mention and the term dependency graph, one can identify all the terms that are similar to the ambiguous mention using stationary distribution for the mention. This means that a random walk algorithm helps identify the related terms. After identifying the related terms, a second graph, which is called context graph for each ambiguous mention is generated with nodes as the related terms and the edges representing the semantic relationship between every node in the graph. The graph is then clustered using one of the three clustering techniques – Louvain, graph based K-means, Hierarchical Agglomerative clustering. These clustering techniques cluster terms related together. In the next entity mapping step, the algorithm takes a list of Wikipedia entity candidates for each ambiguous mention in the tweet. Each generated cluster in the before step, is assigned one Wikipedia Entity from this list using a similarity score. The terms in the cluster are consolidated as a document and the summary of the Wikipedia entity candidate page is considered as a second document. The similarity between these two documents is calculated using one of the three methods- Words Match similarity, UMBC Phrase similarity, UMBC semantic textual similarity [19]. The algorithm calculates similarity scores for each cluster with all the candidate entities and assigns the Wikipedia page with highest similarity score to the cluster. Similar procedure is followed for assigning a Wikipedia Page to all the clusters of the mention. Two are more clusters with the same same Wikipedia page are merged to form a single cluster. The Wikipedia entities assigned to the clusters are the dominant entity candidates for the mention considered. The work shows that the extraction of these few candidate entities which are dominant are enough to improve the performance of entity disambiguation. The method uses these dominant entity candidates in the online step for entity disambiguation in tweets.

The online step involves the actual entity disambiguation process. In this step, the algorithm considers only the dominant entity candidates for the ambiguous mentions in the tweets for disambiguation. For an ambiguous mention in a tweet, one of the entities from dominant candidate list is mapped. The mapping is performed based on the similarity score that is calculated in one of the two ways – Context-based similarity and Collective Similarity. The context-based similarity is calculated using a document similarity score by considering the tweet text as the first document and the entity Wikipedia page summary as the second document. The similarity score is calculated with all the dominant entity candidates and the mention is mapped to the entity with the highest score. The Collective similarity considers all the mentions in a tweet together and mapping is done with the objective to select the entities that are not only similar to the tweet, but also similar to each other. Entities are assigned for all the mentions such that the similarity between entities of different mentions is also high. In this way, disambiguation is performed with few candidate entities.

The experiments show that the method reduces entity linking time and also the effectiveness of dominant candidates in the entity linking process.

2.4 Clustering Methods

In this work, before disambiguation, context extension is performed to form contextual groups of tweets from the tweets in the dataset. Different clustering techniques are used to cluster the tweets in an unsupervised fashion. A brief description about each of the clustering techniques used in this thesis is given below.

2.4.1 K-Means

K-Means [27] is the one of the most popular clustering techniques available. K-Means clustering technique tries to minimize the intra-cluster distances. In other words, the algorithm minimizes the sum of squares of the distances between the points in the cluster and the centroid of the cluster. This is also referred as inertia. This algorithm requires specification of the number of clusters that are expected to be formed by the given dataset which makes it necessary to have knowledge about the data beforehand.

The K-Means algorithm proceeds as follows:

- Centroids are randomly chosen from the dataset. The number of centroids is equal to the number of clusters requested.
- The datapoints are assigned to the cluster that has the centroid near them.
- Once all the points are assigned to their nearest centroid clusters, actual centroids are calculated and reassigned. The distance between the old and new sets of centroids is calculated.

The last 2 steps are repeated until the change in the objective function between two consecutive iterations is less than given tolerance value. K-Means algorithm is usually used for large and medium sized datasets. The algorithm produces even sized clusters.

2.4.2 K-Medoids or K-Medians

K-Medoids clustering technique [22] works similar to K-Means with slight difference. The K-Means algorithm is sensitive to outliers or extreme values. K-Medoids uses median of the points as the cluster center whereas K-Means uses means of the points as the cluster centroid. So, in K-Means cluster center may or may not be the point from the data values but in K-Medoids center of the cluster is one of the points in the dataset.

K-Medoids technique is robust to outliers than K-Means. This is because the cluster centers can be formed correctly and thus influencing the overall clustering of data samples.

2.4.3 Hierarchical Agglomerative Clustering

Hierarchical Clustering is a clustering technique that forms clusters by either top down or bottom up approach. The top down approach starts by considering the whole dataset as a single cluster and splits them to form more clusters. The bottom up approach starts by considering each sample in the dataset as a separate cluster and merges the data samples into clusters.

Hierarchical Agglomerative Clustering (HAC) [8] is a bottom up hierarchical clustering technique. The merge strategy will be based on the linkage criteria used in the algorithm. It is a minimizing criterion that minimizes one of sum of the squared differences within all clusters, the maximum distance between observations of pairs of clusters, the average of the distances between all observations of pairs of clusters or the distance between the closest observations of pairs of clusters.

This clustering can handle large datasets but may become computationally expensive because of the sample variance.

2.4.4 Affinity Propagation

Affinity Propagation [14] is a clustering technique that creates clusters by exchange of messages between pairs of data points. This algorithm does not require the user to specify the number of clusters to form from the data points. This is a major advantage as previous knowledge on data is not required to use the algorithm. The algorithm requires a parameter to specify the number of exemplars to be used.

The affinity propagation algorithm has two types of messages that are passed between the data points. First a responsibility message is sent from one data point to another which is an exemplar, specifying that the second data point should be exemplar for the first one. Second, an availability message from the second point to the first, specifying that the first point should choose the second as its exemplar. The second message is sent after considering all the responsibility messages that the exemplar has received. The algorithm repeats till all the data points are converged.

The algorithm may become complex as the number of samples in the dataset increases. This is because the number of messages that will be exchanged increases and so the time for the communication.

2.4.5 Louvain Clustering

Louvain [4] is also a hierarchical clustering algorithm that performs clustering in a bottom-up fashion. The Louvain clustering is used as a community detection technique in large networks. This algorithm uses modularity as the optimization function. Modularity is a measure of that quantifies that assignment of node to a cluster. Modularity values range between -1 and 1. Higher the modularity of a node with a cluster, there is a greater chance that this node belongs to the cluster.

The Louvain algorithm proceeds as follows:

- Each node is considered as one community or cluster.
- Remove a node from its community and merge it with the community with which it has highest modularity gain.
- Repeat the step before until there is no change in the communities.
- Consider each community as a hyper node and repeat the process.

The Louvain algorithm involves calculating modularity gains between nodes and clusters. So, Louvain method is fast and also complexity grows linearly with number of nodes in the network.

2.4.6 Hybrid Hierarchical K-Means Clustering

As the name suggests, Hybrid Hierarchical K-Means Clustering [7], [15] is a hybrid technique that uses both hierarchical agglomerative and K-Means Clustering methods. Assigning cluster centroids initially is a major task in K-Means algorithm on which the performance of the algorithm depends. There exists K-means++ feature in K-Means that assigns initial centroids randomly. K-Means also offers to repeat the initial centroids assignment multiple times and select one set of centroids based on the inertia in clustering. This algorithm is used to select initial centroids without K-Means++.

The algorithm first uses agglomerative clustering to cluster the data points. The clusters are formed based on the hierarchy. Then centers for these clusters are calculated as the mean of the data points. These centers are now used as initial centers in K-means clustering method.

2.5 Text Vectorization

Machine learning algorithms need numerical features to perform mathematical operations like matrix multiplication, factorization and others. Tweet texts are used in the unsupervised clustering methods and so they need to be converted into vectors in this thesis. Commonly used methods to convert text to numerical data are:

- TF-IDF
- Embeddings

TF-IDF

TF-IDF [51] is a popular method for term-weighting. Term Frequency – Inverse Document Frequency (TF-IDF) represents importance of a word in a document in the corpus or dataset. The two concepts term frequency (TF) and inverse document frequency (IDF) together are used to obtain TF-IDF.

Term Frequency tells us how frequently a word appears in a document. This is calculated with respect to the length of the document as different documents have different length. Term frequency of a word w in a document d is given by

$$TF(w,d) = \frac{\text{Number of times word w appers in document d}}{\text{Total number of words in the document d}}$$

The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs [55]. Inverse document frequency is used to assign more weight to rarely occurring words and less weights of frequently occurring words in a document. Inverse Document Frequency of a word in a corpus is given by

$$IDF(w,C) = \log_{10} \frac{\text{Number of Documents in the corpus C}}{\text{Number of Documents in which the word w is present}}$$

Term Frequency - Inverse Document Frequency is the product of Term Frequency and Inverse Document Frequency.

$$TF - IDF(w,d,C) = TF(w,d) * IDF(w,C)$$

The value of TF-IDF for a word in a document is always greater than or equal to 0. TF-IDF will be high when the term frequency of a word is high in a document and the word appears in very few documents in the corpus. Therefore TF-IDF will filter out most common terms across the corpus.

Embeddings
Word embedding is also a popular word representation that represents words in terms of vectors. Word embeddings capture context of a word in a text document along with semantic similarity among different words in the document. Word2vec introduced in [33] is one of the techniques used to learn and generate word embeddings. Word2vec uses neural network to learn relationship between words in the training corpus. The model is then used to get the vector representation of words in a document or similarity between the words in a document.

2.6 Knowledge Bases

A Knowledge Base contains number of entities and information about these entities. It also contains relations between different entities present in it. Entities in one Knowledge Base also contain information about the same entity in multiple Knowledge Bases. Wikipedia is the most widely known and used knowledge base. Different Knowledge bases are used for Entity linking research.

For entity disambiguation process, an ideal Knowledge base should contain the following properties:

– Availability: Knowledge bases should be publicly available to all so that data can be accessed without any restrictions.
– Readability: Knowledge bases should be human readable as well as machine readable.
– Authenticity: Entities in a knowledge base should be from a verified human source or a related valid entity from another knowledge base.
– Multilingual: A knowledge base should support multiple languages in-order to facilitate or support systems of multiple languages.

We shall now look into Wikidata Knowledge Base in brief as this thesis uses it for Entity Disambiguation.

2.6.1 Wikidata

Wikidata Knowledge Base was launched in 2012 by Wikimedia Foundations. Wikidata Knowledge Base is a publicly available, multilingual knowledge graph. The data can also be used by other Wikimedia projects like Wikipedia. Wikidata uses Resource Description Framework (RDF) data model. Data can be added into Wikidata by both verified users and automated scripts that directly add data from other Knowledge bases. RDF data model of the Wikidata makes it easier for the users, Wikis maintained by Wikimedia foundation and other third-party programs to write, read and process data from the Knowledge base.

Data in Wikidata is focused on items which can be topics, concepts or objects. Information about each item is stored in the form of statements. These statements are in the form of property-value pairs. Following provides description of Wikidata statements, entities, items and properties retrieved from the Wikidata glossary page [58].

Statements are information about an item stored in the item page that contains key-value pairs, for property with one or more entity values. It may optionally contain reference or source for the data and a rank used to distinguish different statements. For example, the informal English statement "milk is white" would be encoded by a statement pairing the property color (P462) with the value white (Q23444) under the item milk (Q8495) [57]. Statements are usually considered as (item, property, value) triplets.

Entity is the data in the Wikidata page which may be an item, a property or a lexeme. Each entity in the Wikidata is uniquely identified by an identifier which has a prefix based on the type. For example, items have the prefix ‘Q’ and properties have the prefix ‘P’. Entities can also be uniquely identified by the combination of label and description in each language. An entity can also be read or accessed directly using the URI “http://www.wikidata.org/entity/ID” where ID is the unique identifier.

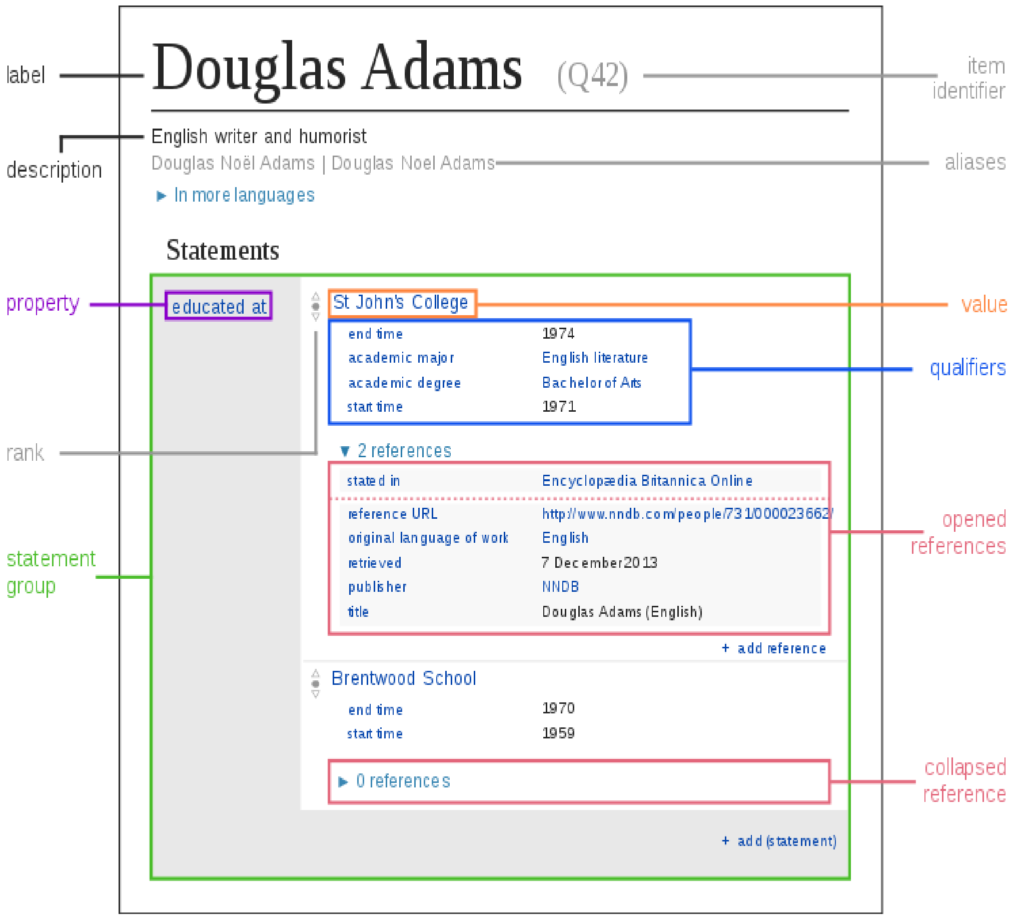
Item in Wikidata refers to a real world entity, concept or an event. An item is uniquely identified by an identifier that has a prefix ‘Q’. For example item with the identifier Q145 refers to the United Kingdom. Each item has a Wikidata page that contains list of statements for that item. Item can also be identified by a site link to an external site, or by unique combination of multilingual label and description. Item is the subject part of the statement or triplet.

Property in Wikidata refers to the data value of the statement. It is also referred as attribute value as these are features of the item that they belong to.

Properties have their own pages on Wikidata and are connected to items, resulting in a linked data structure. Property is the predicate of the statement or triplet. Each property in a statement will be mapped to a single value, set of values or some relation. Sometimes properties will have a missing value indicating that there is no value mapped to that property for that item.

Site links in a Wikidata page for an item provides pages for the item in different Knowledge bases like Wikipedia or YAGO, thus connecting Wikidata with other Knowledge Bases. Data in Wikidata Knowledge Base can be accessed using queries. There are many built-in tools, external tools and programming interfaces to query Wikidata. SPARQL queries are widely used to query Wikidata. Among the properties, the property P31, which gives the type or category of the entity in Wikidata, is mostly queried. The Figure 2.2 shows a Wikidata page of Douglas Adams with details.

As of now, there are about 89 Million entities in the Wikidata and data dumps will be updated every week. With increasing amounts of data items and a structured data model, Wikidata can be considered as a potential Knowledge base for Entity Disambiguation.



Douglas Adams (Q42)

English writer and humorist

Douglas Noël Adams | Douglas Noel Adams

▸ In more languages

Statements

educated at

St John's College

end time	1974
academic major	English literature
academic degree	Bachelor of Arts
start time	1971

▾ 2 references

stated in	Encyclopædia Britannica Online
reference URL	http://www.nndb.com/people/731/000023662/
original language of work	English
retrieved	7 December 2013
publisher	NNDB
title	Douglas Adams (English)

+ add reference

Brentwood School

end time	1970
start time	1959

▸ 0 references

+ add (statement)

Figure 2.2: Data model in Wikidata [13]

Chapter 3

Methodology

This section presents GraphEDM, our Entity Disambiguation framework for microposts. Figure 3.1 summarizes the method we adopt for entity disambiguation. Our approach is divided into two main phases: Context Extension and Graph-Based Disambiguation. Context Extension uses clustering approaches to extend the limited context of microposts, while Graph-Based Disambiguation disambiguates mentions through a graph-based approach leveraging the extended context. Each phase is described in more detail below.

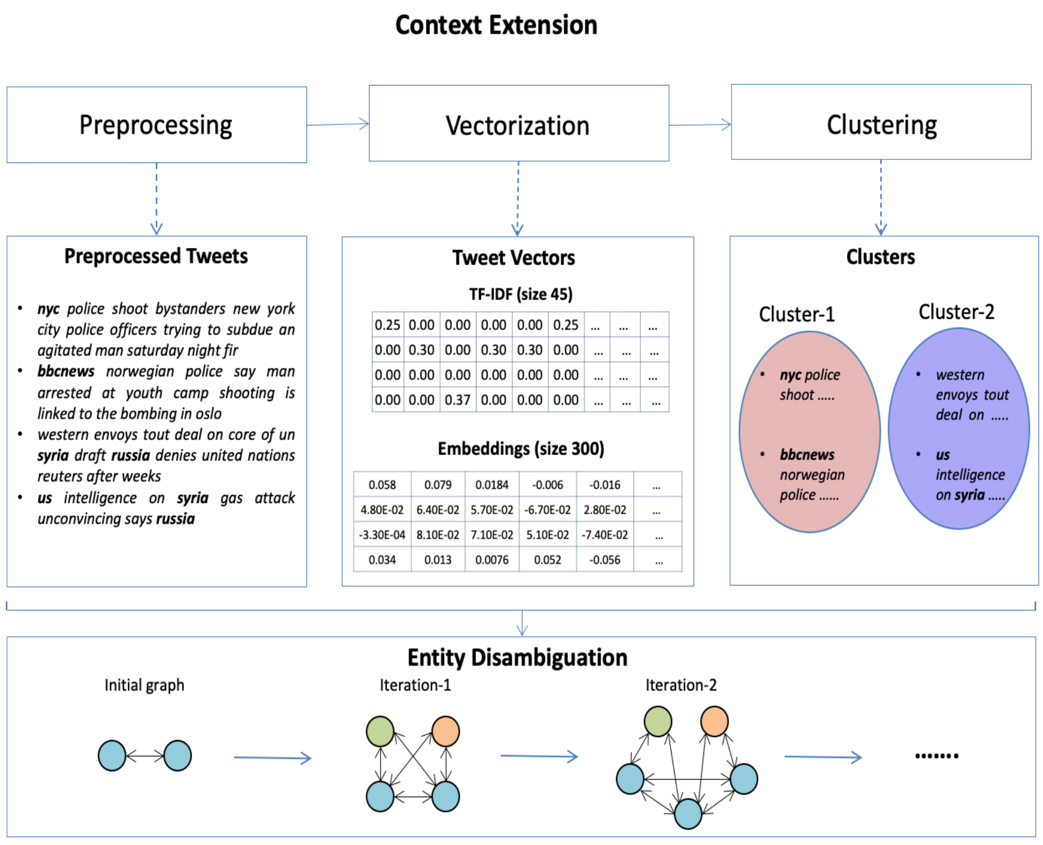
3.1 Context Extension

The context required to correctly disambiguate entities in tweets can often be missing as the text of a tweet can be fragmented, noisy, and very short. To fill this gap, we propose an unsupervised clustering approach to extend the context by borrowing context from related tweets. This process takes place in three steps. First, we preprocess the textual contents of the tweets to align them to the same canonical form. Next, we convert each tweet into a TF-IDF [51] or embedding representation [33]. These vector representations are finally used to generate clusters of related tweets. We explain the process in detail below.

3.1.1 Preprocessing

The text of a tweet is limited to 280 characters, and the content is quite often written in an informal manner with slang, abbreviations, emoticons, URLs, and Hashtags [6], [44]. Thus, preprocessing is a necessary first step before automatic tweet analysis. Below are removed from the tweet in preprocessing.

- Emoticons and punctuation: The tone behind an emoticon is difficult to understand and can be misleading. For clustering the tweets, these emoticons and special character are of no value. We consider these as noise and remove these from the tweet text.



- ***nyc*** *police shoot bystanders new york city police officers trying to subdue an agitated man saturday night fir*
- ***bbcnews*** *norwegian police say man arrested at youth camp shooting is linked to the bombing in oslo*
- *western envoys tout deal on core of un* ***syria*** *draft* ***russia*** *denies united nations reuters after weeks*
- ***us*** *intelligence on* ***syria*** *gas attack unconvincing says* ***russia***

0.25	0.00	0.00	0.00	0.00	0.25	...	...	...
0.00	0.30	0.00	0.30	0.30	0.00	...	...	...
0.00	0.00	0.00	0.00	0.00	0.00	...	...	...
0.00	0.00	0.37	0.00	0.00	0.00	...	...	...

0.058	0.079	0.0184	-0.006	-0.016	...
4.80E-02	6.40E-02	5.70E-02	-6.70E-02	2.80E-02	...
-3.30E-04	8.10E-02	7.10E-02	5.10E-02	-7.40E-02	...
0.034	0.013	0.0076	0.052	-0.056	...

Cluster-1:
- ***nyc*** *police shoot .....*
- ***bbcnews*** *norwegian police ......*

Cluster-2:
- *western envoys tout deal on .....*
- ***us*** *intelligence on* ***syria*** *.....*

Figure 3.1: The complete methodology of our proposed approach, GraphEDM. The approach is divided into two phases, Context Extension, followed by Entity Disambiguation. The words in bold are mentions in a tweet.

- Web URLs and HTML references: Many tweets contain an HTML reference to a page which provides additional information. This reference link may contain an image, a video or even a website which may or may not be about the topic being discussed in the tweet. The context extension step does not use these HTML references in the text for clustering tweets and remove them from the text.
- 'RT' in Retweeets: Twitter allows users to tweet on their own previous tweets or tweets from other people. These tweets are marked with prefix 'RT' at the beginning of the text. In context extension task, there might be a chance that tweets that are retweeted and have this prefix are clustered together which is not desired. So, the method removes this from the text.

Next, we tokenize the tweet text by converting each sentence into individual informative tokens.

3.1.2 Vectorization

Tweet text contains numerical values, characters and strings. To cluster tweets, different types in the tweet text are converted into numerical values. The converted representation should directly correspond to the actual values or should represent a comparison between the actual values. Different techniques for text vectorization have been seen in Section 2.5. In this thesis after preprocessing, Tweets are converted into vectors using TF-IDF and Embeddings.

Tweet TF-IDF
To represent tweets using TF-IDF vectors, we consider each tweet in the dataset as a document and TF-IDF values are calculated for each word in a tweet. To improve the performance of the clustering, stop words are removed from the tweet texts. Hence, each tweet in the dataset has a dimension equivalent to the number of words in the dataset.

Tweet Embeddings
For the embedding representation, a publicly available pre-trained Google Word2vec model was used [1]. The model is trained on the Google news dataset of about 100 billion words, and each word is represented by a vector of 300 dimensions. After cleaning the tweet text, each tweet is split into separate words, and the vector representation of each word is loaded as a vector from the Google pre-trained model. Each tweet is then represented by a vector that is the average of the vectors of the words present in that particular tweet. So, each tweet in the dataset is represented by a vector of 300 dimensions.

3.1.3 Clustering

The final step in the process of generating a more meaningful context for a tweet is clustering. For clustering of tweets, tweet texts are used in the form of vectors generated in the previous step. In machine learning, clustering text is done in an unsupervised manner. Many text clustering techniques exist that are proven to be effective. In this work, K-Means, K-Medoids, Agglomerative, Affinity Propagation, Hybrid Hierarchical K-Means, Louvain clustering techniques are used to cluster the tweets. In K-Means, Agglomerative, HHK-Means clustering, Euclidean distance metric is used. On the other hand, in K-Medoids and Affinity Propagation clustering, cosine similarity is used.

Input for different clustering techniques can vary. For K-Means, K-Medoids, Agglomerative, Affinity Propagation, and Hybrid Hierarchical K-Means, each tweet is considered as one data point for clustering. The output of the clustering is then a set of clusters of semantically related tweets.

In addition, we use the Louvain clustering [4] method, where a set of words in the dataset are considered as nodes in the graph. An edge connects two nodes if they occur in the same tweet. Subsequently, we perform clustering on this graph of words. A tweet is then assigned to the cluster where the majority of the words appearing in its textual content belong. It is important to mention here that clustering is performed directly on the text, and no vectorization approach is used.

Almost all the clustering algorithms require specification of the number of clusters parameter. This feature tells the algorithm the number of clusters to form from the given dataset.

[1] https://code.google.com/archive/p/word2vec/

The clustering algorithm is majorly influenced by this factor. Graphical methods like elbow method, average silhouette method and gap statistic are used to choose the value of this parameter. The methods decide on the value of the parameter by comparing the effect of clustering on the given data points. In this thesis, we use elbow method to decide on the feature.

Figure 3.1 illustrates our methodology for Entity Disambiguation with tweets provided as input. The preprocessing step cleans the tweet text. These preprocessed tweets are then vectorized into TF-IDF or embedding vectors as explained in Section 4.5.3. Further, we use clustering techniques to cluster these tweet vectors based on similarity, as shown in the Figure 3.1. Tweets within the same cluster share semantic similarity. These clusters become the extended context for each tweet inside it. Specifically, our disambiguation algorithm uses all the mentions present in tweets within a cluster. For the example shown in Figure 3.1, 'nyc' and 'bbcnews' are mentions of the tweets within the same cluster and will be used to generate the graph.

3.2 Entity Disambiguation

After the context extension phase of the system, entity disambiguation is performed using the previously generated clusters. We perform this task using a graph-based approach. In the previous step, we obtain clusters of tweets with high semantic similarity within the same cluster. Next, we perform Entity Disambiguation (as illustrated in Figure 3.1) in the following way:

1. First, all the mentions from the tweets within a cluster are fetched together into a list. For each mention in the list, candidate entities are obtained by querying a surface form index. The surface form is a collection of key-value pairs, where the key is a label, and the values are Wikidata identifiers. For example, for the label *New York City*, the surface form outputs the Wikidata identifiers *Q7013143, Q60, Q3875477.*
2. An initial graph is built using unambiguous mentions (obtained in the above step) where all nodes are connected to each other.
3. Considering an ambiguous mention m, a k-partite graph is constructed with all k candidate entities of the mention m.
4. Ambiguity is resolved for the mention m, and added the initial graph.
5. This process is repeated for each ambiguous mention of the cluster iteratively.

Describing Step 4 in more detail, the process of resolving an ambiguous mention m is explained now. First, a Word2vec model is trained using all the candidate entities from the Wikidata Knowledge Base. The Word2vec training dataset is built by extrapolating triplets from the Wikidata knowledge graph and considering each triplet as a sentence. Subsequently, for the k-partite graph constructed above in Step 3, each node is represented using the embedding representation obtained from the trained Word2vec model on Wikidata. For each pair of nodes, the weight of the edge is given by the similarity computed in Equation 3.1:

$$weight(v_1, v_2) = \frac{\cos\left(emb\left(v_1\right), emb\left(v_2\right)\right)}{\sum_k \cos\left(emb\left(v_1\right), emb\left(k\right)\right)} \tag{3.1}$$

where v_1 and v_2 are the nodes representing the mentions and k is a node that has an edge from v_1.

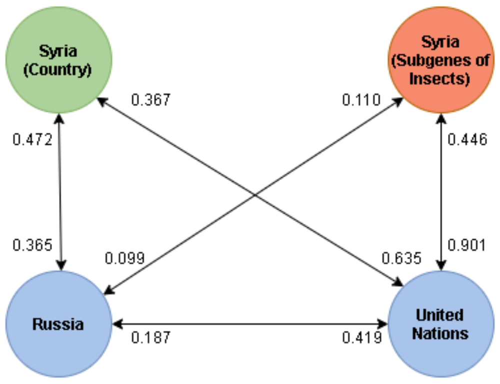


Figure 3.2: Entity disambiguation with our iterative method

The PageRank [39] centrality ranking method is applied to the above graph, and the candidate node with the highest ranking score is the entity that gets assigned to the mention. This mapping is considered to be likely correct, and the mapped entity is added to the initial graph (Step 2 above). The type of the mentions (property: P31) is fetched and the three major types existing among all the mentions are saved. When the centrality score for two or more nodes is same, an entity whose type matches one of the top three types is assigned to the mention. If there are no unambiguous mentions to generate the initial graph, the ambiguous mention in the first iteration is also disambiguated using this saved type information. The number of iterations required depends on the number of ambiguous mentions in the list of mentions. The output of the iterative method is a list of mapped mention-entity pairs.

Figure 3.2 illustrates the iterative method used for disambiguation. The graph contains two nodes - *United Nations* and *Russia*, that are already disambiguated. The current iteration disambiguates the mention *Syria* and considers two candidate entities: *Syria (Country)* and *Syria (Subgenes of insects)*. The centrality ranking algorithm assigns a high ranking score for entity *Syria* with entity type *Country*, and this entity is assigned to the mention. This entity is then added to the graph, and the disambiguation process continues with the next ambiguous mention.

The iterative method maps an entity from the Knowledge Base for each mention in the set. If this method fails to assign an entity to a mention, then it means that the mention is not disambiguated in the process. This happens if there is no entity for the given mention in the Knowledge Base or the page found by the algorithm is either a disambiguation page or a category page. Thus, the output of the disambiguation algorithm is a list of mapped mention-entity pairs. This list corresponds to the cluster from which the mentions list is extracted. So, the number of lists with mention-entity pairs is equal to the number of clusters generated in the context extension phase.

In each cluster, mentions in the tweets are mapped to corresponding entities only from the list corresponding to the cluster. This means that the mentions in the tweets from cluster 1 are mapped to entities in the list 1, mentions in the tweets from cluster 2 are mapped to entities in the list 2 and so on. In this regard, the same mention from 2 different clusters may be mapped to different entities in looping method, if the context in the two clusters is different.

The algorithm assigns an entity to the mention in the tweet in a cluster if a mapping exists for the mention in the list returned by the looping method. If there is no mapping for the mention from the looping method, then the mention is not assigned with any entity which means that entity disambiguation is missing for that mention in the tweet.

Chapter 4

Experiments & Results

In this section, we introduce our datasets, experimental setup, and SoTA baselines. Further, we raise few research questions followed by their answers in the form of experiments and evaluations..

4.1 Datasets

In this thesis, seven different datasets are used that are available publicly. The gold standard datasets used in this thesis are listed below

- Making Sense of Microposts (Micropost 2014) Training set
- Making Sense of Microposts (Micropost 2014) Test set
- Making Sense of Microposts (Micropost 2016) Training set
- Making Sense of Microposts (Micropost 2016) Test set
- Making Sense of Microposts (Micropost 2016) Development set
- Brian Collection
- Mena Collection

These datasets provide Tweet Id and the mentions in the tweet along with the mapped entity from a Knowledge Base. Tweet texts are not publicly available and so needs to be extracted using Tweet API. Depending on the user access, Tweet API provides details about a tweet given a Tweet id. In this thesis, Tweepy Python library [50] is used to extract tweet texts from Twitter for the Tweet Ids present in the datasets. The corresponding text could not be extracted for a number of tweets using the Twitter API; The reason behind this was either that the tweets were deleted from Twitter, or that we could not have access to those due to privacy reasons.

Table 4.1 gives key information about the seven publicly available datasets that we used to evaluate our framework. The table contains the number of tweets, the number of unique entities, and the total number of entities present in each dataset. Each dataset contains tweet ids and a corresponding list of mentions.

Table 4.1: Database Statistics

Dataset	**No. of Tweets**	**Unique Entities**	**Total Entities**
Micropost2014 Test	698	617	1014
Micropost2014 Training	1518	1522	2938
Micropost2016 Test	296	195	737
Micropost2016 Training	4073	2789	6368
Micropost2016 Development	100	98	253
Brian Collection	1603	384	1231
Mena Collection	162	348	482

4.1.1 Making Sense of Microposts (Micropost 2014)

Making Sense of Microposts 2014 is a Named Entity Extraction and Linking challenge organized as part of World Wide Web Conference 2014. The challenge involved tasks of Named Entity Extraction and Linking in Tweets. The statistics on dataset released for the challenge are described in [5]. These datasets are used as baseline in different Tweet entity linking works like [11] and [18]. The statistics of training set and test set used in the thesis are given below. The Micropost 2014 challenge used DBpedia Knowledge Base for entity linking. So the train and test gold standard datasets have DBpedia entities. These entities are mapped from DBpedia to Wikidata for using in this thesis. So, in the gold standard dataset, each tweet has one or more entities with DBpedia page link, whereas the dataset for this thesis contains one or more entities with Wikidata page ID. If there is no Wikidata Page for a DBpedia page, then the corresponding mention-entity pair is removed from the dataset.

Training Set

Training set of Micropost 2014 consists of 2,340 tweets out of which 1,518 tweet texts could be extracted. So, this thesis uses these 1518 tweets for the disambiguation. The dataset contains 6330 tokens. The number of entities in the dataset is 2938 and the number of unique entities is 1522.

Test Set

Test set of Micropost 2014 consists of 1,165 tweets out of which 698 tweet texts could be extracted. The dataset contains 3443 tokens. The number of entities in the dataset is 1014 and the number of unique entities is 617.

4.1.2 Making Sense of Microposts (Micropost 2016)

Making Sense of Microposts (Micropost 2016) is also a NEEL challenge held as part of World Wide Conference 2016. The challenge is on tweets in English language and the dataset contains tweets that are manually annotated. This dataset also has entities mapped from DBpedia Knowledge Base and for this thesis these are mapped on to the corresponding Wikidata pages. Similar to the dataset before, if there is no corresponding Wikidata page for the DBpedia page, the mention-entity pair is removed and not considered for named entity disambiguation. The statistics on dataset released for the challenge are described in [48].

Training Set
Training set of Micropost 2016 consists of 4,073 tweets out of which all the 4,073 tweet texts could be extracted. The dataset contains 13,053 tokens. The number of entities in the dataset is 6368 and the number of unique entities is 2789.

Test Set
Test set of Micropost 2016 consists of 296 tweets out of which all 296 tweet texts could be extracted. The dataset contains 1,431 tokens. The number of entities in the dataset is 737 and the number of unique entities is 195.

Development Set
Development set of Micropost 2016 consists of 100 tweets of which tweet texts could be extracted for all the tweets. The dataset contains 544 tokens. The number of entities in the dataset is 253 and the number of unique entities is 98.

4.1.3 Brian Collection

Brian Collection is used for Named Entity Extraction in [26]. The dataset is extended for Named Entity Linking for use in [18]. The dataset contains 1,603 tweets that are annotated manually. The annotation is done using Wikipedia Knowledge Base. Few non-Wikipedia mention-entity pairs, URLs of websites or related webpages are considered. According to [18], there are 1585 mentions in the dataset out of which 1,233 entities are from Wikipedia, 274 are non-Wikipedia entities and the remaining 78 are the mentions with no matching entity.

In this thesis, this dataset is used to assess the performance of the disambiguation algorithm. The dataset used in the thesis contains all the 1,603 tweets from the original dataset. The dataset contains 4,591 tokens. For this thesis, entities from Wikipedia are only considered and mapped to the corresponding entities in Wikidata Knowledge Base. Mentions that do not have any corresponding entity and non-Wikipedia entities are not considered. The mentions for which Wikipedia entities could not be mapped to a corresponding Wikidata page are also ignored. The number of entities in the dataset is 1231 and the number of unique entities is 384.

4.1.4 Mena Collection

Mena collection is used in [17] for Named Entity Extraction. The dataset is then used in [18] for entity Linking. Entity mapping is similar to the Brian dataset. This dataset contains 162 tweets with 510 mentions. The mentions are mapped to 483 Wikipedia entities, 19 non-Wikipedia entities and rest 8 are not annotated as given in [18].

In this thesis, all the tweets from Mena collection are used for entity disambiguation. The dataset contains 1,289 tokens from all the tweets. The number of entities in the dataset is 482 and the number of unique entities is 348. Similar to Brian Collection, only Wikipedia entities are mapped to Wikidata entities and used.

4.2 Dataset Preparation

The seven datasets listed before are used in this thesis for assessing the performance of the entity disambiguation method. These datasets are taken from different sources and are in different format. Micropost 2014 and Micropost 2016 gold standard datasets contain details like tweet id and mention-entity pairs. Micropost 2014 dataset files are tab separated files (tsv) with each line in the file structured as tweet id, mention1, entity1, mention2, entity2 and so on each separated by a tab space. A mention and its entity are also separated by a tab space. Micropost 2016 dataset files are also tab separated files and are different from Micropost 2014 files. In Micropost 2016 dataset files, each mention-entity pair of a tweet is given in a separate line. So, if there are 3 mentions in a tweet, then the file contains 3 lines with same tweet id and each with a different mention-entity pair separated by a tab space. For the thesis, Brian collection and Mena collection datasets are taken from [18] and that work uses the datasets in XML format. These files contain tweet id, tweet text, mentions and mapped entities as xml tags.

All the datasets needs to be in the same consistent format to use in this thesis. We generate two files for each dataset and both these files are tab separated files. The first file contains the tweet details and mentions that are to be mapped in the disambiguation process and the second file contains the mapped entities. Each line in this file includes a tweet id, its text belonging to the tweet id and comma separated list of mentions (enclosed in square brackets) each separated by a tab space. The datasets also contain tweets that have no mentions in it and for these tweets a "-" is present in place of the list of mentions. The second file serves as a ground truth file for the disambiguation. Each line in this file includes a tweet id and comma separated list of entities for the mentions in the tweet from other file. The entities in the file are Wikidata page ids that start with "Q" and followed by the page id. For a tweet, the number of mentions in the first file and the number of entities in the second file are equal. Similar to list of mentions, the list of entities is also enclosed in square brackets. If the tweet does not have mentions to be mapped, then the tweet is not present in the ground truth file. This means that the number of tweets in the two files is different.

As stated before, dataset preparation also includes mapping of entities from DBpedia and Wikipedia Knowledge Bases to Wikidata Knowledge Base. Entities in the Knowledge Base tend to change frequently. Some of the DBpedia and Wikipedia pages could not be mapped to corresponding Wikidata pages. Example of one of this case encountered is the DBpedia page "`http://dbpedia.org/resource/A.S._Red_&_Blue`". This page does not have a corresponding Wikidata Page. Mentions like this are removed from the datasets to maintain consistency.

4.3 Baselines

We compare the performance of our entity disambiguation framework against three state-of-the-art baselines – AIDA, WAT-API, and ELTDS. We use standard precision, recall, and F1 scores as evaluation metrics for all methods.

4.3.1 AIDA

AIDA [60] is a state-of-the-art method for entity disambiguation. The online tool disambiguates entities in the text that is provided to it. AIDA's performance results for some datasets were obtained from the Gerbil platform[1]. The Gerbil benchmarking platform is web-based and developed as part of [49]. It provides annotation results for different registered annotators for a predefined list of datasets. Gerbil was used to get the results for the Micropost2014 and 2016 datasets. For the Brian and Mena Collections, experimental results from [18] were used where experiments are performed on these two datasets with AIDA

4.3.2 WAT-API

WAT-API is another state-of-the-art method that provides better performance than TagMe [12], a well-known annotator for short-texts. The corresponding API provides multiple services like NEEL and D2KB computing Entity Relatedness, among other features. To compare WAT-API to our proposed method, we used the D2KB service of the API. Requests were sent to the D2KB service with the tweet texts and mentions. The response contained mentions and their mapped entities along with other information from the disambiguation algorithm of WAT-API. We report the results obtained using this service in the tables below.

The service provides an URL that needs an access token. The token is issued to a user when registered to the service. HTTP GET request can be sent using the provided URL and tweet details for D2KB service. A JSON object with two keys is added to the URL. The first one is "text" parameter that contains tweet text and the second is "suggested_spans" that contains the list of mention's start index and end index. Indices of multiple mentions can be provided in the JSON object to be sent in the same request. The response contains mentions and their mapped entities along with other information from disambiguation algorithm of WAT-API. More information about using this service is provided in [41]. For each tweet in the dataset, one HTTP GET request is sent and response is obtained. Mention-entity pairs in response are collected and checked for correctness. Once all the tweets are disambiguated using this tool, precision, recall and F1 score are calculated for all the tweets in the dataset together.

4.3.3 ELTDS

ELTDS [11] uses a training corpus of almost 8 Million tweets to find the dominant candidates entities. For each ambiguous mention in the tweets, the algorithm forms term clusters with the related terms from the tweets and maps each cluster to a candidate entity of the mention. The candidate entities mapped onto these clusters are identified as the dominant entity candidates

[1]http://gerbil.aksw.org/gerbil/

and used for disambiguation. We used the publicly available code to obtain the performance of ELTDS on all our datasets.

The system implementation is in Python and MySQL database is used to store data. Below are missing from the implementation available and so are created for this thesis baseline.

- Database: As part of the thesis, for this baseline, SQLITE database is opted to use with the Python code. Changes are made in the code to change MySQL queries to SQLITE queries. SQLITE database is created for the project and necessary tables are created. Data needed for the code to run is inserted and consistency is maintained in the database.
- Mention Dictionary: Mention dictionary is needed by the algorithm to identify potential mentions in the tweets. This is not available with the code. The mention dictionary used in the paper implementation has about 28000 mentions. For this purpose, a mention dictionary is created to be used in the implementation that contains about 12000 mentions.
- Related terms file: A file containing list of related words for each mention in the mention dictionary is also not available. Related terms for a mention are obtained using the term clusters as described in Section 2.3.2.2. As seen, random walk algorithm is used on the term cluster graph to get a list of related words for a mention. The file is prepared for the mentions in the mention dictionary previously created.

The missing parts of the implementation are implemented and the entity disambiguation is performed on all the tweets in all the datasets. The performance measures, precision, recall and F1 score are calculated as in the original implementation.

4.4 Experiments

The proposed GraphEDM method is implemented in Python. Experiments are conducted for all the available datasets for the proposed disambiguation method and baselines.

The tweets texts are converted into vectors using both TD-IDF approach and Embeddings. For TF-IDF, TfidfVectorizer from scikit-learn is used and for embeddings Word2vec is used. All the clustering algorithms described in Section 2.4 are used on all the datasets and then entity disambiguation is performed on the clusters generated. The clustering algorithms from scikit-learn Python package is used to cluster the tweets. As seen before, the clustering algorithms need a parameter to be passed that is the number of clusters to generate from the datasets. This parameter is decided using elbow method for K-Means clustering technique. The Figure 4.1 is an example of a graph generated by elbow for Micropost 2014 Test dataset that used embeddings for tweet vectorization. The number of clusters opted for this clustering technique with the help of elbow method is four. Graphs are generated for all the datasets similarly.

As in the work [10], Wikidata local endpoint server is setup using Blazegraph to be used for entity disambiguation. The steps to set up this endpoint are described in [54]. In this thesis, the Wikidata dump used is the 2019 version.

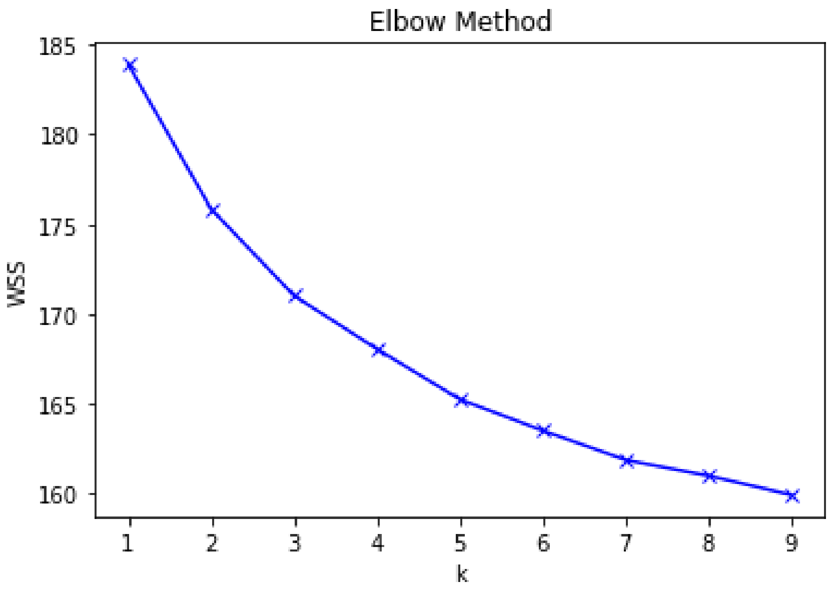


Figure 4.1: Elbow method for Micropost 2014 Test dataset

4.5 Research Questions

Our empirical validation focuses on three research questions:

- **Q1:** How does our proposed Entity Disambiguation framework perform when compared against existing state-of-the-art methods for disambiguating entities in tweets?
- **Q2:** What is the effect of the various clustering techniques used in the Context Extension phase on performance?
- **Q3:** What is the effect of the various vectorization techniques used in the Context Extension phase on the performance?

We answer each of those questions below.

4.5.1 Results - Q1

Table 4.2 shows the results of our proposed framework GraphEDM compared against state-of-the-art baselines for Entity Disambiguation. K-Means clustering technique and Word2vec embeddings for vectorization are used for the results shown in Table 4.2. It is important to mention in this context that the results from our framework can be improved by varying the clustering or vectorization methods see below Section 4.5.2 and Section 4.5.3. As observed, our proposed method outperforms all baseline methods on five datasets and are very competitive on the other two datasets.

The precision of WAT-API is very high for all datasets compared to other systems. This is because WAT-API has a separate step for pruning that increases its precision. In this step, the system removes all the mention-entity pairs that have been disambiguated and are non-coherent with the tweet or text under consideration. Our method results in a high recall compared to other methods as the candidate generation method used in this thesis that uses a surface form index can retrieve more relevant entities.

Below a short analysis on the results for all the datasets individually:

- On the Micropost 2014 Training, Test, Micropost 2016 Test, Development and Brian datasets, our proposed method shows better results than the baselines chosen for comparison. Though WAT-API's precision is high, its low recall results in an overall subpar F1 score.
- On the Micropost 2016 Training dataset, the F1 metrics of GraphEDM is better than AIDA and ELTDS. WAT-API performs better when compared to our method for F1. On this dataset, our approach suffered, as many mentions were not mapped onto an entity. This is mainly due to the fact that those mentions differ from the surface forms of the corresponding entities in the Knowledge Base. The recall of our system is better than that of the baselines while our F1 score is only 3% below that of WAT-API.
- On the Mena collection, the results of our method are better than AIDA and ELTDS. The F1 score of our approach, which is 76.6% is competitive to the F1 score of the WAT-API method, which is 77.6%.

Table 4.2: Performance evaluation of GraphEDM against SoTA baselines

Dataset	Method	Precision	Recall	F1 score
Micropost2014 Test	AIDA	-	-	41.2
	WAT-API	80.6	31.8	45.6
	ELTDS	48.4	32.1	38.6
	GraphEDM	53.3	51.8	**52.5**
Micropost2014 Training	AIDA	-	-	50.3
	WAT-API	83.3	39.2	53.3
	ELTDS	54	30.4	38.9
	GraphEDM	56.5	55.3	**55.9**
Micropost2016 Test	AIDA	-	-	19.2
	WAT-API	77.6	28.7	41.9
	ELTDS	71.8	48.3	57.8
	GraphEDM	61.3	60.5	**60.9**
Micropost2016 Training	AIDA	-	-	48.5
	WAT-API	81.8	47.5	**60.1**
	ELTDS	47.9	33.4	39.6
	GraphEDM	57.4	56.6	57
Micropost2016 Development	AIDA	-	-	15.3
	WAT-API	90.6	32.1	47.4
	ELTDS	72.4	50.8	59.7
	GraphEDM	62.4	61.7	**62**
Brian Collection	AIDA	50.05	29.4	37.04
	WAT-API	86.4	46	59.2
	ELTDS	40.6	40	40.3
	GraphEDM	60	59.9	**60**
Mena Collection	AIDA	72.63	55.69	63.04
	WAT-API	92.9	66.6	**77.6**
	ELTDS	79.7	47.9	59.8
	GraphEDM	76.6	76.6	76.6

4.5.2 Results - Q2

Experiments are conducted with multiple clustering techniques for the Context Extension stage of our framework. Tables 4.3 - 4.9 show the results of our GraphEDM framework for each dataset using various clustering techniques. The clustering techniques used in this sense are K-Means, K-Medoids, Hierarchical Agglomerative Clustering (HAC), Affinity Propagation, Hybrid Hierarchical K-Means (HHK-Means), and Louvain Clustering. Below is the analysis of results with respect to each dataset.

- **Micropost 2014 Training dataset**: Table 4.3 shows the performance metrics for this dataset. The F1 score for all the clustering techniques range between 54.6% and 57.8%. The least score of 54.6 is obtained with Agglomerative clustering and the highest score is with Louvain clustering technique.
- **Micropost 2014 Test dataset**: Table 4.4 shows the performance metrics for this dataset. The F1 score for all the clustering techniques range between 52.1% and 53.1%. The least score of 52.1 is obtained with hybrid hierarchical K-Means clustering using embeddings and the highest score is also with the same clustering technique but when TF-IDF is used to represent tweets.
- **Micropost 2016 Training dataset**: Table 4.5 shows the performance metrics for this dataset. The F1 score for all the clustering techniques range between 57.8% and 55.8%. The best score is obtained from HHK-Means and the least score is from Louvain method.
- **Micropost 2016 Test dataset**: Table 4.6 shows the performance metrics for this dataset. The F1 score for all the clustering techniques range between 59.7% and 61.9%. The best score of 61.9 is obtained when Agglomerative clustering is used with embeddings. The least score of 59.7% is seen with Affinity propagation technique.
- **Micropost 2016 Development dataset**: Table 4.7 shows the performance metrics for this dataset. The F1 score for all the clustering techniques range between 57.7% and 63.2%. The highest and the least scores are obtained with Affinity Propagation and HHK-Means clustering techniques respectively.
- **Brian Collection**: Table 4.8 shows the performance metrics for this dataset. The F1 score for all the clustering techniques vary largely ranging between 51% and 64.1%. The highest score is obtained with K-Means clustering technique and the least score with Agglomerative clustering method.
- **Mena Collection**: Table 4.9 shows the performance metrics for this dataset. The F1 score for all the clustering techniques range between 76.3% and 73.9%. K-Means and Affinity propagation clustering methods both have the same score that is best. The least score is from HHK-Means clustering method.

All clustering techniques used in this thesis are unsupervised, and single technique may not be optimal on different datasets. As described above, the performance differ depending on the data at hand, with HHK-Means, a hybrid clustering technique combining Agglomerative and K-Means clustering, performing best overall and reaching top scores on two different datasets.

4.5.3 Results - Q3

We experimented with two different vectorization techniques in our method, TF-IDF and Word2vec-based embeddings. Each clustering technique is used in combination with both embeddings and TF-IDF vectors. The performance of the system with both vectorization techniques is reported in Tables 4.3 - 4.9. Louvain clustering is unique as it considers each word in the tweet dataset as a node in the graph, and no vectorization method is used for this clustering technique.

We observe that the performance of our system varies depending on the vectorization method used. This was expected as the two method are quite different: Word2vec embeddings are based on the distributional hypothesis and define words through their linguistic contexts, while TF-IDF considers the words frequencies in the documents and corpus, hence creating very different vectors. The number of cluster parameters provided to the clustering techniques is also different in both cases. Out of the seven datasets, the best score is obtained using TF-IDF vectors for three datasets. For two other datasets, the clustering methods using embeddings are best. One dataset shows the same performance with embeddings and TF-IDF vectors, while Louvain reaches the best score on the last dataset.

On average, our results are better with embeddings. In large datasets, TF-IDF vectors can be sparse, which leads to affecting the context extension phase negatively. On the other hand, embeddings are trained on a large amount of dataset and are very effective at representing the context.

Table 4.3: GraphEDM - Micropost 2014 Training dataset

Clustering	Vectorization	Pr	Re	F1
K-Means	Embeddings	56.5	55.3	55.9
	TF-IDF	55.9	54.8	55.3
K-Medoids	Embeddings	57.3	56.2	56.8
	TF-IDF	56	54.8	55.3
HAC	Embeddings	57.6	56.5	57
	TF-IDF	55.5	54.4	55
Affinity - Propagation	Embeddings	55.1	54	54.6
	TF-IDF	-	-	-
HHK-Means	Embeddings	56.6	55.5	56
	TF-IDF	55.7	54.6	55.1
Louvain	-	57.3	56.2	**57.8**

Table 4.4: GraphEDM - Micropost 2014 Test dataset

Clustering	Vectorization	Pr	Re	F1
K-Means	Embeddings	53.3	51.8	52.5
	TF-IDF	53.8	52.3	53
K-Medoids	Embeddings	53.4	51.9	52.6
	TF-IDF	53.5	52.1	52.8
Hierarchical	Embeddings	53.8	52.3	53
Agglomerative	TF-IDF	53.5	52.1	52.8
Affinity Propagation	Embeddings	53.1	51.6	52.3
	TF-IDF	-	-	-
HHK-Means	Embeddings	52.8	51.4	52.1
	TF-IDF	53.8	52.4	**53.1**
Louvain	-	53.7	52.2	52.9

Table 4.5: GraphEDM - Micropost 2016 Training dataset

Clustering	Vectorization	Pr	Re	F1
K-Means	Embeddings	58.3	57.4	**57.8**
	TF-IDF	57.6	56.7	57.2
K-Medoids	Embeddings	56.4	55.6	56
	TF-IDF	57.6	56.7	57.1
Hierarchical	Embeddings	57.5	56.6	57.1
Agglomerative	TF-IDF	57.7	56.8	57.2
Affinity Propagation	Embeddings	-	-	-
	TF-IDF	-	-	-
HHK-Means	Embeddings	57.9	57	57.5
	TF-IDF	58.2	57.3	**57.8**
Louvain	-	56.3	55.4	55.8

Table 4.6: GraphEDM - Micropost 2016 Test dataset

Clustering	Vectorization	Pr	Re	F1
K-Means	Embeddings	61.3	60.5	60.9
	TF-IDF	61.8	61.3	61.6
K-Medoids	Embeddings	61.8	61.3	61.6
	TF-IDF	61.4	60.9	61.2
Hierarchical	Embeddings	62.1	61.6	**61.9**
Agglomerative	TF-IDF	59.5	59	59.2
Affinity Propagation	Embeddings	59.9	59.4	59.7
	TF-IDF	-	-	-
HHK-Means	Embeddings	60.3	59.8	60.1
	TF-IDF	61.3	60.8	61
Louvain	-	61.8	61.3	61.6

Table 4.7: GraphEDM - Micropost 2016 Development dataset

Clustering	Vectorization	Pr	Re	F1
K-Means	Embeddings	62.4	61.7	62
	TF-IDF	60	59.3	59.6
K-Medoids	Embeddings	60.4	59.7	60
	TF-IDF	61.2	60.5	60.8
Hierarchical	Embeddings	60	59.3	59.6
Agglomerative	TF-IDF	60	59.3	59.6
Affinity Propagation	Embeddings	58.4	57.7	58.1
	TF-IDF	63.6	62.8	**63.2**
HHK-Means	Embeddings	58	57.3	57.7
	TF-IDF	58.4	57.7	58.1
Louvain	-	62.8	62.1	62.4

Table 4.8: GraphEDM - Brian Collection

Clustering	**Vectorization**	**Pr**	**Re**	**F1**
K-Means	Embeddings	60	59.9	60
	TF-IDF	64.2	64.1	**64.1**
K-Medoids	Embeddings	59.3	59.2	59.3
	TF-IDF	63.5	63.4	63.5
Hierarchical	Embeddings	61.4	61.3	61.3
Agglomerative	TF-IDF	50.9	51	51
Affinity Propagation	Embeddings	-	-	-
	TF-IDF	-	-	-
HHK-Means	Embeddings	59.6	59.5	59.6
	TF-IDF	60.9	60.8	60.8
Louvain	-	56.6	56.5	56.5

Table 4.9: GraphEDM - Mena Collection

Clustering	**Vectorization**	**Pr**	**Re**	**F1**
K-Means	Embeddings	76.6	76.6	**76.6**
	TF-IDF	76.1	76.1	76.1
K-Medoids	Embeddings	75.9	75.9	75.9
	TF-IDF	75.9	75.9	75.9
Hierarchical	Embeddings	75.7	75.7	75.7
Agglomerative	TF-IDF	74.3	74.3	74.3
Affinity Propagation	Embeddings	76.3	76.3	76.3
	TF-IDF	75.7	75.7	75.7
HHK-Means	Embeddings	73.9	73.9	73.9
	TF-IDF	74.3	74.3	74.3
Louvain	-	75.7	75.7	75.7

Chapter

5

Conclusion

In this thesis, GraphEDM, an effective framework for entity disambiguation in microposts is proposed. The proposed method is composed of two phases. To extend the limited context given by the tweets, clustering techniques are used, to regroup semantically similar messages. Then, disambiguation of the mentions is performed using an iterative graph-based approach. Our proposed approach outperforms the SoTA by up to 15.13% on five out of the seven gold standard datasets in the field of entity disambiguation in tweets.

In future work, we plan to study the effect of the dataset's size on the clustering and on the corresponding results. We believe that our clustering could benefit from more content in order to consolidate the tweets contexts. Also, we would be interested in exploring how our method could be used to process tweets in real time; Specifically, we plan to experiment with dynamic clustering methods where a tweet can be dynamically assigned to existing pretrained clusters to make the disambiguation process more efficient for real-time event detection tasks. Finally, we would like to selectively enhance the surface form matching (for example by leveraging surface forms from additional Knowledge Bases) for the cases where we cannot find any matching candidates in order to optimize recall.

Acknowledgements

Firstly, I would like to thank my husband, my parents and my in-laws for the love and support throughout my Masters including this Master Thesis.

I would like to sincerely thank my supervisors Akansha Bhardwaj and Paolo Rosso for their insights, continuous support and feedback throughout my thesis. I would also thank Prof. Philippe Cudré-Mauroux for providing me an opportunity to do thesis with the Department of Informatics, University of Fribourg.

Finally I would like to thank my friends for making my masters more fun-filled.

Appendix

A

Appendix

The results of all the conducted experiments are tabulated in this section. Each table provides performance of each clustering technique for each vectorization method for all the datasets. As seen from the tables, the parameter number of clusters is varied for each dataset. Affinity Propagation and Louvain clustering methods do not take this number of clusters parameter as input and the number of clusters depends on the algorithm. So there are no tables for Affinity Propagation and Louvain clustering methods. Complete Named Entity Extraction and Linking (NEEL) is performed with the GraphEDM for disambiguation and WAT-API for entity extraction. The performance of NEEL is tabulated in the Table A.9.

Table A.1: Performance of GraphEDM with K-Means clustering using Embeddings

Dataset	**No. Of Clusters**	**Precision**	**Recall**	**F1Score**
Micropost2014 Test	3	52.99	51.53	52.25
	4	53.29	51.82	52.55
	5	52.99	51.53	52.25
	6	52.89	51.43	52.15
Micropost2014 Training	6	55.21	54.12	54.66
	18	56.46	55.34	55.90
Micropost2016 Test	3	61.26	60.52	60.89
	4	60.74	60.24	60.49
	7	61.01	60.52	60.76
Micropost2016 Training	15	56.87	55.99	56.43
	20	57.45	56.56	57.00
	50	57.71	56.84	57.28
	55	57.76	56.89	57.33
	60	58.27	57.40	57.83
	65	57.40	56.53	56.96
	70	58.21	57.33	57.77
	80	57.56	56.69	57.12
	100	57.97	57.10	57.53
Micropost2016 Development	3	62.40	61.66	62.03
	5	58.40	57.71	58.05
Brain Collection	5	57.69	57.60	57.64
	6	52.16	52.07	52.11
	15	59.97	59.87	59.92
Mena Collection	2	73.65	73.65	73.65
	3	75.73	75.73	75.73
	4	76.35	76.35	76.35
	5	75.73	75.73	75.73
	6	75.10	75.10	75.10
	7	76.35	76.35	76.35
	8	75.10	75.10	75.10
	10	76.56	76.56	76.56
	15	75.10	75.10	75.10
	20	75.73	75.73	75.73
	22	75.93	75.93	75.93

Table A.2: Performance of GraphEDM with K-Means clustering using TF-IDF

Dataset	**No. Of Clusters**	**Precision**	**Recall**	**F1Score**
Micropost2014 Test	4	53.80	52.32	53.05
	6	53.50	52.02	52.75
	8	53.09	51.63	52.35
Micropost2014 Training	30	55.56	54.46	55.00
	40	55.87	54.77	55.31
Micropost2016 Test	3	61.83	61.33	61.58
	4	61.15	60.65	60.90
Micropost2016 Training	70	57.64	56.74	57.19
	100	56.53	55.60	56.06
Micropost2016 Development	3	59.60	58.89	59.24
	4	58.80	58.10	58.45
	5	60.00	59.29	59.64
	6	59.20	58.50	58.85
	7	59.60	58.89	59.24
	20	56.80	56.13	56.46
Brian Collection	3	49.55	49.47	49.51
	5	64.20	64.09	64.15
	7	49.88	49.80	49.84
	15	63.55	63.44	63.50
Mena Collection	3	73.65	73.65	73.65
	4	76.14	76.14	76.14
	5	73.86	73.86	73.86
	6	73.86	73.86	73.86
	15	75.10	75.10	75.10
	20	75.73	75.73	75.73
	25	73.44	73.44	73.44

Table A.3: Performance of GraphEDM with K-Mediods clustering using Embeddings

Dataset	No. Of Clusters	Precision	Recall	F1Score
Micropost2014 Test	4	52.68	51.23	51.95
	5	53.09	51.63	52.35
	6	53.39	51.92	52.65
	8	52.68	51.23	51.95
Micropost2014 Training	5	55.69	54.59	55.14
	6	57.33	56.19	56.75
	8	55.80	54.70	55.24
Micropost2016 Test	3	61.01	60.52	60.76
	4	61.29	60.79	61.04
	5	61.83	61.33	61.58
	6	61.56	61.06	61.31
Micropost2016 Training	20	56.43	55.55	55.99
Micropost2016 Development	2	58.40	57.71	58.05
	3	58.40	57.71	58.05
	4	58.40	57.71	58.05
	5	60.40	59.68	60.04
Brian Collection	2	47.68	47.60	47.64
	3	57.85	57.76	57.80
	4	56.55	56.46	56.50
	5	50.85	50.77	50.81
	6	50.20	50.12	50.16
	15	59.32	59.22	59.27
	17	54.43	54.35	54.39
Mena Collection	3	75.31	75.31	75.31
	4	74.07	74.07	74.07
	6	74.27	74.27	74.27
	7	75.93	75.93	75.93
	8	75.10	75.10	75.10
	12	74.90	74.90	74.90

Table A.4: Performance of GraphEDM with K-Mediods clustering using TF-IDF

Dataset	No. Of Clusters	Precision	Recall	F1Score
Micropost2014 Test	4	53.50	52.07	52.77
Micropost2014 Training	40	55.90	54.80	55.35
Micropost2016 Test	3	61.42	60.92	61.17
Micropost2016 Training	60	57.02	56.16	56.58
	70	57.57	56.71	57.14
	75	57.45	56.53	56.99
	80	57.00	56.14	56.57
Micropost2016 Development	3	58.40	57.71	58.05
	4	60.00	59.29	59.64
	5	58.40	57.71	58.05
	20	61.20	60.47	60.83
	26	56.40	55.73	56.06
Brian Collection	3	48.58	48.50	48.54
	4	50.94	50.85	50.89
	5	48.90	48.82	48.86
Mena Collection	2	75.93	75.93	75.93
	3	75.10	75.10	75.10
	4	74.07	74.07	74.07
	5	73.65	73.65	73.65

Table A.5: Performance of GraphEDM with Agglomerative clustering using Embeddings

Dataset	**No. Of Clusters**	**Precision**	**Recall**	**F1Score**
Micropost2014 Test	2	53.19	51.72	52.45
	3	53.80	52.32	53.05
	4	53.80	52.32	53.05
	6	52.79	51.33	52.05
Micropost2014 Training	6	56.25	55.14	55.69
	18	57.60	56.47	57.03
Micropost2016 Test	3	59.78	59.29	59.54
	4	59.92	59.43	59.67
	5	60.05	59.57	59.81
	7	62.11	61.60	61.85
	9	61.56	61.06	61.31
Micropost2016 Training	15	57.53	56.63	57.08
	20	57.40	56.51	56.95
	25	56.80	56.13	56.46
Micropost2016 Development	3	58.40	57.71	58.05
	4	58.00	57.31	57.65
	5	58.40	57.71	58.05
	7	58.40	57.71	58.05
	15	58.00	57.31	57.65
	20	60.00	59.29	59.64
	25	56.80	56.13	56.46
Brian Collection	7	57.85	57.76	57.80
	6	61.35	61.25	61.30
	5	58.10	58.00	58.05
	4	50.28	50.20	50.24
	15	53.95	53.86	53.90
Mena Collection	4	75.73	75.73	75.73

Table A.6: Performance of GraphEDM with Agglomerative clustering using TF-IDF

Dataset	No. Of Clusters	Precision	Recall	F1Score
Micropost2014 Test	3	53.50	52.07	52.77
	4	52.99	51.58	52.27
	5	53.09	51.68	52.37
	6	52.99	51.58	52.27
Micropost2014 Training	30	55.49	54.39	54.93
Micropost2016 Test	3	59.51	59.02	59.26
Micropost2016 Training	200	53.79	37.64	44.29
	300	57.68	56.82	57.25
Micropost2016 Development	3	58.00	57.31	57.65
	4	60.00	59.29	59.64
	5	58.40	57.71	58.05
	15	58.40	57.71	58.05
	20	58.80	58.10	58.45
	30	58.80	58.10	58.45
Brian Collection	3	50.37	50.28	50.33
	4	49.72	49.63	49.67
	5	50.12	50.04	50.08
	6	50.53	50.45	50.49
	7	50.94	50.85	50.89
	10	47.52	47.44	47.48
	15	50.04	49.96	50.00
	30	51.02	50.93	50.98
Mena Collection	4	73.86	73.86	73.86
	6	73.24	73.24	73.24
	8	74.27	74.27	74.27
	10	73.44	73.44	73.44
	12	73.86	73.86	73.86

Table A.7: Performance of GraphEDM with Hybrid Hierarchical K-Means clustering using Embeddings

Dataset	No. Of Clusters	Precision	Recall	F1Score
Micropost2014 Test	3	52.08	50.69	51.37
	4	52.79	51.38	52.07
Micropost2014 Training	18	56.56	55.45	56.00
Micropost2016 Test	3	60.33	59.84	60.08
Micropost2016 Training	40	57.32	56.45	56.88
	50	57.91	57.04	57.47
	70	57.91	57.04	57.47
	75	57.76	56.89	57.33
	90	57.81	56.94	57.37
Micropost2016 Development	3	58.00	57.31	57.65
	4	58.00	57.31	57.65
Brian Collection	5	49.39	49.31	49.35
	8	51.99	51.91	51.95
	15	59.64	59.55	59.59
Mena Collection	3	73.86	73.86	73.86
	5	76.14	76.14	76.14
	6	75.31	75.31	75.31
	7	74.69	74.69	74.69

Table A.8: Performance of GraphEDM with Hybrid Hierarchical K-Means clustering using TF-IDF

Dataset	**No. Of Clusters**	**Precision**	**Recall**	**F1Score**
Micropost2014 Test	3	52.99	51.58	52.27
	4	53.80	52.37	53.07
	5	53.50	52.02	52.75
Micropost2014 Training	40	55.66	54.56	55.10
Micropost2016 Test	3	55.66	54.56	55.10
Micropost2016 Training	300	58.20	57.32	57.75
Micropost2016 Development	3	58.40	57.71	58.05
	2	58.00	57.31	57.65
	4	58.40	57.71	58.05
Brian Collection	5	60.86	60.76	60.81
Mena Collection	2	73.86	73.86	73.86
	3	74.27	74.27	74.27
	4	73.86	73.86	73.86
	5	74.07	74.07	74.07

Table A.9: Performance of GraphEDM along with entity extraction with WAT-API

Dataset	**No. Of Clusters**	**Precision**	**Recall**	**F1Score**
MSME 2014 Test	5	59.63	26.67	36.86
MSME 2014 Train	18	59.18	32.63	42.07
MSME 2016 Test	3	52.6	19.61	28.57
MSME 2016 Train	20	59.56	43.1	50.01
MSME 2016 Dev	3	55.06	23.33	32.78
Brian Collection	5	58.64	50.17	54.08
Mena Collection	3	79.8	59.21	67.98

Bibliography

[1] R. Alrashdi and S. O'Keefe, "Automatic labeling of tweets for crisis response using distant supervision," in *Companion Proceedings of the Web Conference 2020*, 2020, pp. 418–425.

[2] F. Atefeh and W. Khreich, "A survey of techniques for event detection in twitter," *Computational Intelligence*, vol. 31, no. 1, pp. 132–164, 2015.

[3] A. Bhardwaj, A. Blarer, P. Cudré-Mauroux, V. Lenders, B. Motik, A. Tanner, and A. Tonon, "Event detection on microposts: A comparison of four approaches," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, P10008, 2008.

[5] A. E. Cano, G. Rizzo, A. Varga, M. Rowe, M. Stankovic, and A.-S. Dadzie, "Making sense of microposts:(# microposts2014) named entity extraction & linking challenge," in *CEUR Workshop Proceedings*, vol. 1141, 2014, pp. 54–60.

[6] S. Carter, W. Weerkamp, and M. Tsagkias, "Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text," *Language Resources and Evaluation*, vol. 47, no. 1, pp. 195–215, 2013.

[7] B. Chen, P. C. Tai, R. Harrison, and Y. Pan, "Novel hybrid hierarchical-k-means clustering method (hk-means) for microarray analysis," in *2005 IEEE Computational Systems Bioinformatics Conference-Workshops (CSBW'05)*, IEEE, 2005, pp. 105–108.

[8] L. Despalatović, T. Vojković, and D. Vukicevic, "Community structure in networks: Girvan-newman algorithm improvement," in *2014 37th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, IEEE, 2014, pp. 997–1002.

[9] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, and V. Christophides, "Matching web tables with knowledge base entities: From entity lookups to entity embeddings," in *International Semantic Web Conference*, Springer, 2017, pp. 260–277.

[10] Y. Eslahi, A. Bhardwaj, P. Rosso, K. Stockinger, and P. Cudré-Mauroux, "Annotating web tables through knowledge bases: A context-based approach," in *2020 7th Swiss Conference on Data Science (SDS)*, IEEE, 2020, pp. 29–34.

[11] Y. Feng, F. Zarrinkalam, E. Bagheri, H. Fani, and F. Al-Obeidat, "Entity linking of tweets based on dominant entity candidates," *Social Network Analysis and Mining*, vol. 8, no. 1, p. 46, 2018.

[12] P. Ferragina and U. Scaiella, “Tagme: On-the-fly annotation of short text fragments (by wikipedia entities),” in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 1625–1628.

[13] “File:datamodel in wikidata.svg,” [Online]. Available: `https://commons.wikimedia.org/wiki/File:Datamodel_in_Wikidata.svg`. (accessed: 03.09.2020).

[14] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[15] A. Ganpati and J. Sharma, “A hybrid implementation of k-means and hac algorithm and its comparison with other clustering algorithms,”

[16] M. J. Garbade, *A simple introduction to natural language processing*. [Online]. Available: `https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32`.

[17] M. B. Habib and M. Van Keulen, “Unsupervised improvement of named entity extraction in short informal context using disambiguation clues.,” in *SWAIE*, 2012, pp. 1–10.

[18] M. B. Habib and M. Van Keulen, “Twitterneed: A hybrid approach for named entity extraction and disambiguation for tweet,” *Natural language engineering*, vol. 22, no. 3, p. 423, 2016.

[19] L. Han, A. L. Kashyap, T. Finin, J. Mayfield, and J. Weese, “Umbc_ebiquity-core: Semantic textual similarity systems,” in *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, 2013, pp. 44–52.

[20] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. De Melo, and G. Weikum, “Yago2: Exploring and querying world knowledge in time, space, context, and many languages,” in *Proceedings of the 20th international conference companion on World wide web*, 2011, pp. 229–232.

[21] H. Huang, Y. Cao, X. Huang, H. Ji, and C.-Y. Lin, “Collective tweet wikification based on semi-supervised graph regularization,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 380–390.

[22] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[23] N. Kolitsas, O.-E. Ganea, and T. Hofmann, “End-to-end neural entity linking,” *arXiv preprint arXiv:1808.07699*, 2018.

[24] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee, “Twiner: Named entity recognition in targeted twitter stream,” in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 2012, pp. 721–730.

[25] X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu, “Entity linking for tweets,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 1304–1311.

[26] B. Locke and J. Martin, “Named entity recognition: Adapting to microblogging,” Senior Thesis, University of Colorado, 2009.

[27] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[28] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, “The stanford corenlp natural language processing toolkit,” in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[29] E. Marsh and D. Perzanowski, “Muc-7 evaluation of ie technology: Overview of results,” in *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*, 1998.

[30] E. Meij, W. Weerkamp, and M. De Rijke, “Adding semantics to microblog posts,” in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 563–572.

[31] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, “Dbpedia spotlight: Shedding light on the web of documents,” in *Proceedings of the 7th international conference on semantic systems*, 2011, pp. 1–8.

[32] R. Mihalcea and A. Csomai, “Wikify! linking documents to encyclopedic knowledge,” in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 233–242.

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

[34] B. Molina. “Twitter overcounted active users since 2014, shares surge on profit hopes. - usa today.,” [Online]. Available: `https://eu.usatoday.com/story/tech/news/2017/10/26/twitter-overcounted-active-users-since-2014-shares-surge/801968001/`. (accessed: 01.09.2020).

[35] A. D. D. Munoz, R. M. Unanue, A. P. Garcıa-Plaza, and V. Fresno, “Unsupervised real-time company name disambiguation in twitter,” in *ICWSM Workshop on Real-Time Analysis and Mining of Social Streams*, 2012, pp. 25–28.

[36] G. von Nordheim, K. Boczek, and L. Koppers, “Sourcing the sources,” *Digital Journalism*, vol. 6, no. 7, pp. 807–828, 2018. DOI: `10.1080/21670811.2018.1490658`. eprint: `https://doi.org/10.1080/21670811.2018.1490658`. [Online]. Available: `https://doi.org/10.1080/21670811.2018.1490658`.

[37] R. Nordquist, *Syntactic ambiguity.* [Online]. Available: `https://www.thoughtco.com/syntactic-ambiguity-grammar-1692179`.

[38] R. Otto, *Transfer learning for named entity linking with deep learning*, 2018.

[39] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web.,” Stanford InfoLab, Tech. Rep., 1999.

[40] T. Pellissier Tanon, G. Weikum, and F. Suchanek, “Yago 4: A reason-able knowledge base,” *The Semantic Web*, 2020.

[41] F. Piccinno, *Wat api documentation.* [Online]. Available: `https://sobigdata.d4science.org/web/tagme/wat-api`.

[42] F. Piccinno and P. Ferragina, “From tagme to wat: A new entity annotator,” in *Proceedings of the first international workshop on Entity recognition & disambiguation*, 2014, pp. 55–62.

[43] A. Pipitone, G. Tirone, and R. Pirrone, “Named entity recognition and linking in tweets based on linguistic similarity,” in *Conference of the Italian Association for Artificial Intelligence*, Springer, 2017, pp. 101–113.

[44] D. Ramachandran and R Parvathi, “Analysis of twitter specific preprocessing technique for tweets,” *Procedia Computer Science*, vol. 165, pp. 245–251, 2019.

[45] C. Ran, W. Shen, and J. Wang, “An attention factor graph model for tweet entity linking,” in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1135–1144.

[46] J. Ray Chowdhury, C. Caragea, and D. Caragea, “Keyphrase extraction from disaster-related tweets,” in *The world wide web conference*, 2019, pp. 1555–1566.

[47] A. Ritter, S. Clark, O. Etzioni, *et al.*, “Named entity recognition in tweets: An experimental study,” in *Proceedings of the 2011 conference on empirical methods in natural language processing*, 2011, pp. 1524–1534.

[48] G. Rizzo, M. V. Erp, J. Plu, and R. Troncy, “Making sense of microposts (#microposts2016) named entity recognition and linking (neel) challenge,” in *#Microposts*, 2016.

[49] M. Röder, R. Usbeck, and A.-C. Ngonga Ngomo, “Gerbil–benchmarking named entity recognition and linking consistently,” *Semantic Web*, vol. 9, no. 5, pp. 605–625, 2018.

[50] J. Roesslein, “Tweepy,” *Python programming language module*, 2015.

[51] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[52] W. Shen, J. Wang, P. Luo, and M. Wang, “Linden: Linking named entities with knowledge base via semantic knowledge,” in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 449–458.

[53] D. Spina, E. Amigó, and J. Gonzalo, “Filter keywords and majority class strategies for company name disambiguation in twitter,” in *International Conference of the Cross-Language Evaluation Forum for European Languages*, Springer, 2011, pp. 50–61.

[54] M. Stanislav, Dupont, and D. James Earl, *Wikibase rdf query*, 2018. [Online]. Available: `https://github.com/wikimedia/wikidata-query-rdf/blob/master/docs/getting-started.md`.

[55] “Tf–idf - wikipedia,” [Online]. Available: `https://en.wikipedia.org/wiki/Tf%E2%80%93idf`. (accessed: 08.09.2020).

[56] “Twitter - wikipedia,” [Online]. Available: `https://en.wikipedia.org/wiki/Twitter`. (accessed: 01.09.2020).

[57] "Wikidata - wikipedia," [Online]. Available: `https://en.wikipedia.org/wiki/Wikidata\Property_and_value`. (accessed: 03.09.2020).

[58] "Wikidata:glossary," [Online]. Available: `https://www.wikidata.org/wiki/Wikidata:Glossary`. (accessed: 03.09.2020).

[59] C. Xu, J. Li, X. Luo, J. Pei, C. Li, and D. Ji, "Dlocrl: A deep learning pipeline for fine-grained location recognition and linking in tweets," in *The World Wide Web Conference*, 2019, pp. 3391–3397.

[60] M. A. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum, "Aida: An online tool for accurate disambiguation of named entities in text and tables," *Proceedings of the VLDB Endowment*, vol. 4, no. 12, pp. 1450–1453, 2011.

[61] S. Zwicklbauer, C. Seifert, and M. Granitzer, "Doser-a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings," in *European Semantic Web Conference*, Springer, 2016, pp. 182–198.

Acronyms

BOW	Bag-Of-Words.
EL	Entity Linking.
HAC	Hierarchical Agglomerative Clustering.
IE	Information Extraction.
KB	Knowledge Base.
NED	Named Entity Disambiguation.
NEE	Named Entity Extraction.
NEEL	Named Entity Extraction and Linking.
NER	Named Entity Recognition.
NERC	Named Entity Recognition and Classification.
NLP	Natural Language Processing.
RDF	Resource Description Framework.
TF-IDF	Term Frequency – Inverse Document Frequency.