Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Incident Linking: Assigning Tweets to Entries in a Disaster Database

Master's Thesis

Siva Bathala

1. Referee: Prof. Dr. Benno Stein
2. Referee: Prof. Dr.-Ing. Volker Rodehorst

Submission date: June 25, 2021

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, June 25, 2021

. . . . . .
Siva Bathala

Abstract

Twitter offers outstanding potential for collecting information about disaster events. However, these disaster events usually are recorded in incident databases. Consequently, we want to extract the structured data from tweets to supplement the entries in the traditionally acquired incident databases if some of the metadata (impact, location, description, time) is missing and create new entries, if necessary. To accomplish this, there is a need for a linking method that allows analyzing the relation between tweet texts and incident metadata. Unfortunately, established algorithms for event detection and entity linking cannot solve this problem. Event detection algorithms require a large volume of tweets for each event, which is often unavailable, and they do not attempt to connect tweets and incident databases. Entity linking algorithms require named entities in the databases, which are often missing in event databases. A new Incident Linking Framework (ILF) is proposed and evaluated in this thesis to accomplish this task automatically. This method is a two-step approach that includes individual components like candidate generation and candidate ranking. The candidate generation step returns the possible incidents that match a tweet. The candidate ranking ranks these candidates individually using scoring metrics and produces the best-fit pair of tweets and incidents. There are two different experiments conducted to evaluate this approach. Results for ILF shows that tweets can automatically get assigned to the incident database only if it matches the metadata. In addition, the candidate generation shows promising results compared to the candidate ranking.

Acknowledgements

This thesis became a reality with the kind support and help of many individuals. I want to take this opportunity and extend my sincere gratitude to all of them.

I am honored to be a part of the study program at **Bauhaus-Universität Weimar**. During the course of the program, I have been graced with several opportunities to learn new concepts and integrate my existing learning to research projects offered.

I would first like to thank **Prof. Dr. Benno Stein** for letting me work with his department "Web-Technology and Information Systems Group".

I would especially like to take this opportunity to thank my supervisor **M.Sc. Matti Wiegmann** and **Dr. Jens Kersten** from DLR, Jena. They have played a vital role during the entire thesis. They always managed time and had the grit to discuss problems and progress during the whole thesis. Their constant help and guidance during the whole thesis were incomparable.

I am incredibly thankful to our former chief minister of Andhra Pradesh, India **Sri. Nara Chandrababu Naidu** Garu for providing me financial support for my master's degree under the **NTR Videshi Vidya** scheme.

I would also like to thank the **"Federal Ministry of Education and Research (BMBF)"** for providing me financial support during the corona time.

Special mention to my friend Nagaratna Marihal for her assistance not only with english but moral support. Lastly, I would like to thank the rest of my friends, the CS4DM family and my family back home for being there throughout and supporting me whenever needed.

Finally, I would like to dedicate this work to **Sri. Nara Chandrababu Naidu** Garu and **Sri. Nara Lokesh** Garu.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Many disasters and extreme events are happening daily throughout the world. These disasters are an unwanted occurrence caused by forces largely beyond human control that strike rapidly and without notice, threatening severe disruption of life and property, including death and injury to a large number of people (Lone and Subramani, 2016). Over the previous decade, more than 300 natural disasters are harming millions and costing billions annually worldwide (Prasad and Francescutti, 2017). Although these disasters are unpredictable, there are early warning systems that can anticipate them. For instance, we monitor volcanic activity, measure water levels, precipitation, and flood predictions. With these measurements, we can detect or predict few disasters (like floods, storms). To provide a foundation for various analyses over space and time, detected incidents and corresponding observations (incident metadata) are stored in disaster or incident databases. Several databases are available which record such incidents. For example, in the case of earthquakes, we have United States Geological Survey (USGS). It collects, monitors, analyzes, and disseminates scientific information regarding natural resource conditions, challenges, and concerns. For storm, we have NOAA's National Centers for Environmental Information (NCEI), which documents storms and other significant weather events that are severe enough to cause death, injury, considerable property damage, and other noteworthy meteorological phenomena that occur in conjunction with another event, such as record maximum or low temperatures or precipitation.

Similarly, there are a large number of databases from where we can access the data. However, we cannot usually validate the impact of some of those detected (or predicted) disasters. Satellite-based products can provide a clue where this happened. However, we sometimes have to wait several days to get a new image from the area of interest. We could exploit social media to fill these gaps and to obtain information that is not visible on a satellite image. With the increased use of social media, people are now using it to share information or opinions through

posts. These social media posts can be an additional source of data to extract the details of an event's impact.

Twitter is one of the most widely used social media platforms for people to express themselves and share information. According to Whitney (2020) there are 152 million people who use Twitter daily, especially when there is a crisis in the locality. They use hashtags and social media to spread emergency information constantly. According to a survey on Hurricane Sandy (Baer, 2012), people were seen communicating more regularly using social media. People were finding assistance effectively and efficiently as they attempted to reach friends and family in and out of the disaster area in need of updates on transportation, lodging, and food. This makes it possible to manage a natural disaster more efficiently with the large flow of information via social media (Kabir and Madria, 2019).

According to Wiegmann et al. (2021), many different data sources generate incident database entries, like USGS and NCEI. A human has to search for information in a manual process to validate and add database entries. Social media are one common source to support this. The main open issue still is that there are no reliable and general methods available to automatically identify messages from social media containing valuable incident metadata that could be added to the database or be utilized to validate incident database entries. In this thesis, we present an approach to link social media posts with the incident database to validate and fill missing metadata entries in the incident database. As an example, a database entry (shown in Figure 1.1) could be validated by the tweet shown in Figure 1.2. We extract the structure information like location, impact, disaster type, and timestamp from the tweet. We use only the corresponding related details from the incident database to link those tweets with the incident entries. This linking should also be possible in case of slightly varying values (in other words for the same incident, uncertain place mentions, or different numbers of injuries and deaths reported). The proposed method can serve as a foundation for extracting other database-specific metadata that is not covered here.

```
{
  "incident_id": "47108fe1-5c04-472c-b534-75a51b747489",
  "type": "landslide",
  "start_time": "2011-12-08T08:00:00.000Z",
  "end_time": "NaN",
  "location": "Colombia ; Bosa  ;  Bogota",
  "lat": "4.6176",
  "lon": "-74.1899",
  "source_database_id": "2",
  "properties": {
    "id": "4,089",
    "landslide_": "Mudslide",
    "trigger": "Downpour",
    "storm_name": "nan",
    "fatalities": "6",
    "injuries": "0",
    "source_nam": "nan",
    "source_lin": "http://cnsnews.com/news/article/mudslide-collapses-bus-colombia-6-dead",
    "location_a": "Known_within_1_km",
    "landslide1": "Medium",
    "photos_lin": "nan",
    "cat_src": "glc",
    "countrynam": "Colombia",
    "near": "Soacha",
    "distance": "5.1765",
    "adminname1": "Cundinamarca",
    "adminname2": "nan",
    "population": "313,945",
    "countrycod": "nan",
    "continentc": "SA",
    "key_": "CO",
    "version": "1",
    "user_id": "1",
    "tstamp": "Tue Apr 01 2014 00:00:00 GMT+0000 (UTC)",
    "changeset_": "1"
  }
}
```


Figure 1.1: Sample incident database entry from EM-DAT

Figure 1.2: Sample disaster related Tweet

The objective of this thesis is to provide a framework to identify those tweets that can potentially be linked and then try to find the incident in a database that fits the best. Based on this goal, the following research questions are addressed.

> *RQ 1: What are the possible features that we can extract from tweets that match with those of typical knowledge databases?*
>
> *RQ 2: How can we build a linking model that will link the each tweet to entries in the disaster database based on the features from RQ1?*
>
> *RQ 3: How accurate this model to use for disaster linking?*

We propose a framework with three components: Pre-processing and classification, candidate generation, and candidate ranking. The pre-processing and classification module takes tweets and identifies potentially crisis-relevant tweets, i.e., filtering. The candidate generation takes tweets and incidents and gives a list of incidents that match the tweets, and the candidate ranking does the ranking. We evaluated all three on datasets with 23,673 tweets and 23,723 incidents. Pre-processing and classification module achieved an F1 score of 0.84, and the candidate generation module achieved a recall up to 0.89 and best Mean Reciprocal Rank (MRR) of 0.1972 for candidate ranking. The remainder of this thesis is structured as follows.

In Chapter 2, we overview the related studies of linking frameworks such as Entity Linking, Record Linkage, and Knowledge Base Population (KBP). This literature review provides a solid framework for continued research in this areas. In Chapter 3, a new methodology for incident linking is proposed. We also provide an answer to RQ1 and RQ2 in this chapter. Details on the used data sets, the utilized evaluation metrics and the general experimental settings are presented in Chapter 4. We show the results of each component of the proposed methods, and we provide an answer to RQ3 Chapter 5. In Chapter 6, we present the research findings and propose possible further research directions.

Chapter 2

Related Work

This chapter overviews the various fields related to this thesis. The most relevant related fields are Entity Linking, Record Linking, and Knowledge Base Population (KBP). The Incident Linking Framework (ILF) presented in this work is based on multiple concepts used to connect documents or web queries to knowledge-bases.

2.1 Entity Linking

Entity Linking is an Information Extraction task that connects name entity mentions in a web text to their knowledge-base (KB) entries (Rosales Mendez et al., 2018). Information extraction, Information Retrieval, Content Analysis, and knowledge base population are all possible applications. However, this challenge is difficult due to name variants and entity uncertainty (Shen et al., 2015).

The main entity linking system will take named entity mentions M in given text documents and set of Entities E in the Knowledge base as in input, and it will link the named entity mention $m \in M$ to its corresponding entity $e \in E$ in the knowledge base. If there are no linkable mentions identified, then that will be tagged as NIL (a label that states that there is no corresponding entity in the knowledge base) (Shen et al., 2015). The sample workflow of the Entity Linking system shown in Figure 2.1.

Shen et al. (2015) summarized that there is a three-step approach to solve the general Entity Linking task. These steps are given below.

- Candidate generation
- Candidate ranking
- Unlinkable mention prediction

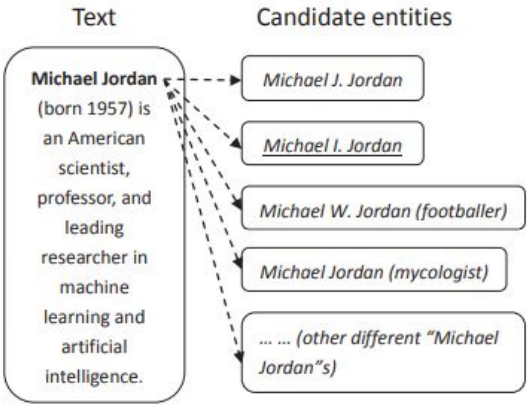

Figure 2.1: An illustration for the entity linking task. The named entity mention detected from the text is in bold face; the correct mapping entity is underlined (Shen et al., 2015).

2.1.1 Candidate generation

Candidate generation is the process to retrieves candidate entities from the knowledge base, based on the mentions in text documents. In the entity linking system, the candidate generation module will consider for each entity mention $m \in M$, and it will include all the possible entities that m may belong to set the candidates list E_m. To make sure that the target entity can appear in the candidate, (Fang et al., 2020) optimized the recall the candidate entities as precisely as possible.

There are many methods used to generate the candidates. All these Candidate generation approaches are mostly focused on string comparisons between the surface form of the entity mention and the name of the entity in a knowledge base (Shen et al., 2015). According to (Hachey et al., 2013) research for any successful entity linking system candidate generation module is critical and as important as the candidate ranking module. Shen et al. (2015) mentioned all the main methods to generate the candidates list which includes Name Dictionary Based Techniques, Surface Form Expansion From The Local Document.

Name Dictionary Based Techniques: A dictionary D is a set of key-value pairs (k,v), where K represents the list of names, and the value column represents its corresponding set of named entities. Author (Shen et al., 2015) has taken Wikipedia as an example to construct the dictionary using a set of features provided by Wikipedia. These features include entity pages, redirect pages, disambiguation pages, bold phrases from the first paragraphs, and hyperlinks in Wikipedia articles. Below Figure 2.2 shows the example of the dictionary-based method.

After constructing the dictionary, the simplest method for generating the candidates E_m for an entity mention $m \in M$ is to check for the exact match or partial

k (Name)	*k.value* (Mapping entity)
Microsoft	*Microsoft*
Microsoft Corporation	*Microsoft*
Michael Jordan	*Michael Jordan* *Michael I. Jordan* *Michael Jordan (footballer)* *Michael Jordan (mycologist)* ...
Hewlett-Packard Company	*Hewlett-Packard*
HP	*Hewlett-Packard*
Bill Hewlett	*William Reddington Hewlett*

Figure 2.2: Sample part of the dictionary where K column represents the name and value column represents the Named entities

between the name k in the key column and the entity mention m. If $k = m$, the set of entities $k.value$ are attached to the candidate entity set E_m (Shen et al., 2015).

Few methods address the misspelling issue in the entity mention before matching with the dictionary, which is very crucial and needs addressed specifically (Shen et al., 2015). Varma et al. (2008) identified spelling variations for a given entity mention using the Metaphone algorithm (Deorowicz and Ciura, 2005). Chen et al. (2010) used Lucene's spellchecker to obtain the suggested correct string.

Surface Form Expansion From The Local Document: Some entity mentions may contain acronyms. Surface Form Expansion methods check for the expanded variations from the local documents where the entity mentions appears. Then these methods could use these surface expansion forms for candidate generation using the dictionary-based methods. These methods can be categorized into heuristic-based methods and supervised learning methods (Shen et al., 2015).

The approaches proposed by Chen et al. (2010) and Lehmann et al. (2010) extend the entity mention in the form of an acronym by checking the textual meaning surrounding the entity mention using heuristic pattern matching. An acronym in parenthesis adjacent to the extension (e.g., Hewlett-Packard (HP)) and an expansion in parenthesis adjacent to the acronym (e.g., UIUC (the University of Illinois at Urbana-Champaign)) are the most similar patterns they used.

The issue with the approach as mentioned earlier (Heuristic Based Methods) is that it will not recognize the complicated acronyms like swapped or missed acronym letters (e.g., "CCP" for "Communist Party of China" and "DOD" for "United States Department of Defense") (Shen et al., 2015). (Zhang et al., 2011) introduced a supervised learning algorithm for locating extended forms for complex acronyms, which improved accuracy of the module.

Methods Based on Search Engines: These methods will use web search engines to recognize candidate entities by exploiting the entire web content (such as

Google). Han and Zhao (2009) used the Google API to apply the entity mention along with a brief context and selected only Wikipedia pages as candidate entities. Furthermore, the Wikipedia search engine is used to find candidate entities and will return a list of related Wikipedia entity pages (Shen et al., 2015).

2.1.2 Candidate ranking

We discussed methods for generating the candidate entity set E_m for each entity listed in the previous section. Now our problem is to rank the candidate entities in E_m and get the top entity from E_m. In any entity linking system, all components are crucial. Still, if our candidate generation step fails, we will not get the candidate entity in the list that we are looking for in the candidate ranking. According to (Shen et al., 2015), these candidate ranking approaches can be divided into supervised and unsupervised ranking methods.

Supervised ranking methods: To "learn" how to rank the candidate entities in E_m, these approaches require annotated training data. Binary classification techniques, learning to rank methods, and graph-based approaches are all examples of these approaches.

Pilz and Paaß (2011) solved the entity ranking problem with binary classification methods. The binary classifier is used to rank if an entity mention is related to a candidate entity for given pair of a candidate entity and an entity mention. If there are two are more candidates identified, Vector Space Model based methods are employed to rank those candidate entities (Zhang et al., 2010).

Chen and Ji (2011) elucidated learning to Rank Methods which rank framework assigns a rank to the entity set and considers relationships between candidate entities for the same entity mention rather than treating them separately as the binary classifier does. Learning to rank is a class of supervised approaches whose purpose is to create a ranking model from training data automatically.

A graph-based model creates a graph for all mentions and associated candidate entities. The example graph-based method shown in Figure 2.3 (Chen and Ji, 2011). A node-set of the graph represents the mentions in the document and all candidate entities of each mention. The weight of the edge is defined based on the cosine similarity. Another way of ranking the candidate entities is given by calculating the degree of the candidate entity node (Wu et al., 2018).

Unsupervised ranking methods: These approaches are built on unlabeled corpora and do not enable the model to be manually annotated. Methods based on the Vector Space Model (VSM) are an example of these techniques.

This is a simple way to rank candidate entities using the unsupervised Vector Space Model (Salton et al., 1975). These methods will avoid manually annotating

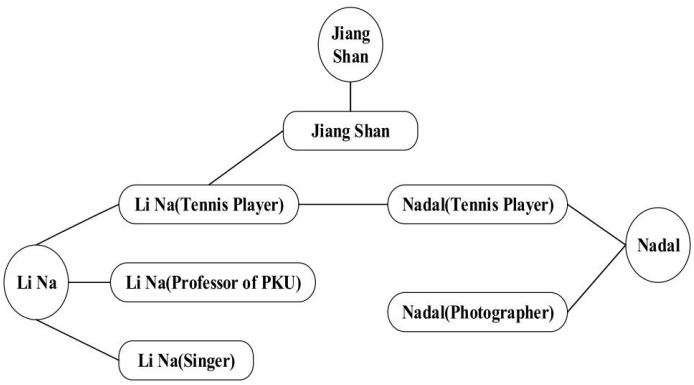

Figure 2.3: Graph based method example where each mention node (circle) has an edge with its candidate entity (ellipse)

the training data, which is more expensive(Chen et al., 2010). These methods calculate the similarity between the vectorial representation of the entity mention and the candidate entity. If the selected candidate entity achieves the highest similarity score, then it is selected as the mapping entity of entity mention (Wu et al., 2018).

2.1.3 Unlinkable Mention Prediction

In the above sections, we went through the most commonly used candidate ranking techniques to generate candidate set E_m. Many methods like (Cucerzan, 2007), contain mapping for all mapping entities for entity mentions. Some systems need to deal with problems like predicting unlinkable mentions if there is a no matching entities in the knowledge base. (Han and Zhao, 2009) proposed a method to predict the unlinkable mention. This system has maintained a certain threshold for NIL which is unlinkable mention, and top score entity is added with a specific score. If the score is smaller than the given threshold, it will return as a NIL; otherwise, it maps the entity to the mention.

The concept for linking entities is used in the thesis in an extended manner, since we try to link information extracted from texts with structured data in databases.

2.2 Record Linkage

Record linkage is the process of matching the identical records across data sources that belong to the same entity. It has many names like deduplication, entity matching. The fundamental part of all record linkage techniques is comparing two values

for the same field to determine if the values match: for example, comparing the first name between two records. Wiegand and Goerge (2019) elucidated the record linkage systems into deterministic and probabilistic linkage systems.

2.2.1 Deterministic record linkage

The deterministic record linkage employs several logical criteria to identify matches in the two records depending on the comparability between different data items. The simple deterministic algorithm compares unique values between the two record items. If two records match, we can consider that as a valid match. Otherwise, it is an invalid match. To make it more complex by allowing the partial match between the unique records by using Jaro-Winkler distance (Wiegand and Goerge, 2019). Hagger-Johnson et al. (2015) has proposed a three-step approach to solve the deterministic algorithm approach based on the combination of different fields in the data set. Research on record linkage has shown a growing interest in recent decades but research on deterministic linkage systems are very less compared to other linkage systems.

In our proposed method, we do not employ any deterministic approach because we do not have any unique values in the data set to match.

2.2.2 Probabilistic record linkage

In probabilistic record linkage, match weight is allocated to each pair of the record. Thus, higher weights indicate a greater probability that the pair is an actual match. When identifiers agree, the match weight contributes positively; otherwise, it contributes to the weight penalty. In the most straightforward instance, each identifier contributes individually to matching weight by considering the discriminatory value of each identifier. For example, gender contributes weight when compared to Date of Birth (Harron et al., 2017). Record linkage can find high-quality matches that the deterministic record linkage matching method would not have achieved (Wiegand and Goerge, 2019). According to cut-off criteria in match weight, the record pairings are categorized as links or as non-links. Often choosing the threshold, pairs of weights over the highest threshold chosen as links, pairs of weights below the bottom threshold as non-links. The ones which fall under the middle of these thresholds require a manual review (Harron et al., 2017). It is vital to select threshold values as modifying thresholds alters the balance between false matches and missed matches (Krewski et al., 2001). According to Dusetzina et al. (2014), selecting the best thresholds is not easy and is typically a subjective process based on manual inspection of record pairings, which are managed by plotting the distribution of match weights.

Many linkage systems typically use a mixture of deterministic and probability

techniques. Deterministic approaches are computationally cheaper compared with probabilistic approaches and are easy to implement (Harron et al., 2017).

2.2.3 Machine learning-based record linkage

Apart from these two record linkages methods, there have been several attempts on machine learning-based record linkage methods. The first approach related to machine learning in record linkage was introduced in 2003 by (Elfeky et al., 2003). An unsupervised clustering and probabilistic methods are compared with the training classifier. Even though the trained classifier provided more reliable results than the other methods in this study, the authors highlight the challenge of obtaining training data. The primary constraint to supervised machine learning techniques is the difficulty of preparing enough annotated training data. For unsupervised machine learning, the obtained results have to be inspected and interpreted manually. In addition, a new training dataset must be developed for each new reference data set incorporated in the system (Gschwind et al., 2019).

However, as discussed in the literature, a pre-trained classifier needed to adopt the machine learning approaches. We do not require a training classifier in our methods. So we do not employ these machine learning-based methods. We will adopt only couple of methods from probabilistic methods in our system.

2.3 Knowledge base population

Knowledge Base Population (KBP) is an NLP task for creating or extending a knowledge base (Glass and Gliozzo, 2018). Text Analysis Conference (TAC) an annual series of open technology evaluations conducted by the National Institute of Standards and Technology (NIST) in 2010. The KBP track of TAC fosters the development of systems capable of extracting data from unstructured text to populate an existing knowledge base or to build a cold start knowledge base (create from scratch) (Getman et al., 2018). The TAC KBP track comprises numerous tasks such as entity discovery, linking, and slot filling (Adel, 2018). Slot filling is a KBP version that searches through the document collection and supplies particular missing information (slots) with appropriate values (Glass and Gliozzo, 2018). Further in this section, we will discuss the knowledge graph, sub-tasks of KBP, and recent literature in KBP. We may also produce a knowledge graph to represent the knowledge in the knowledge database. In this regard, the knowledge base system adds the nodes and edges to the graph. Node represents the mention in a text and the edges represent the relation between the nodes (Glass and Gliozzo, 2018).

Glass and Gliozzo (2018) have divided the KBP system divided into several subtasks and also suggested not to train these systems independently. The subtask of

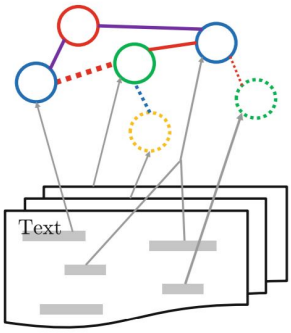

Figure 2.4: Knowledge graph from the text Glass and Gliozzo (2018)

KBP includes Entity Detection and Linking (EDL), Context Set Construction (CSC), relation prediction, and reasoning. The following figure illustrates the association of these components.

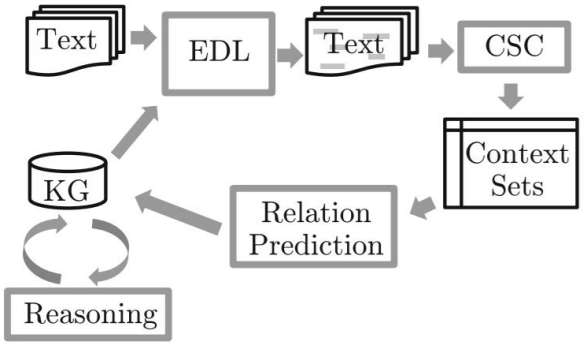

Figure 2.5: Knowledge graph from Glass and Gliozzo (2018)

Entity Detection and Linking (EDL) again divided into three different subtasks: entity detection, co-reference, and entity linking. The main aim of this task is to identify the mentions of the nodes and link them to the knowledge base. Context Set Construction (CSC) system collects textual evidence for the relation prediction phase, recognizing contexts in documents where two-node mention transpires together. Relation prediction and Reasoning the final sub task, which incorporates structured knowledge to predict the new triples.

The KBP task is also indicated as an automated knowledge base and population (Glass and Gliozzo, 2018). Recent academic study has gained substantial interest from the automated knowledge base and population. Since the size of the knowledge is kept expanding, the necessity for automatic building and population of knowledge bases emerged. Existing knowledge bases usually are incredibly unfin-

ished (Asgari-Bidhendi et al., 2021). Therefore, automatic construction of knowledge bases from scratch, populating them with missing information, and adding new knowledge has attracted academic attention.

Asgari-Bidhendi et al. (2021) has proposed the FarsBase Knowledge Base Population system, which consists of state-of-the-art modules such as an entity linking module and information and relation extraction modules for the Persian language. In addition to that, canonicalization was introduced to link extracted relations to FarsBase properties. Canonicalization is the task of mapping the sentences in plain text into predicates in the knowledge base. The canonicalization quality of the KBP system can directly impact its quality. To minimize the human intervention, they have used knowledge fusion techniques. The results obtained suggest that, it works mainly with separate extractor components, the precision of knowledge extraction increased by utilizing a fusion module.

Chapter 3

Proposed Approach

In this chapter, we establish the central approach for the Incident Linking Framework. We introduce the main idea of the incident linking method and steps, including classification, preprocessing, candidate generation, and candidate ranking.

3.1 Incident Linking Framework

For this thesis, we have acquired a microblogs stream of tweets, and the incident database is an event archive taken from online portals based on different timeframes from 2011 to 2019 (see the Chapater 4). The primary method of our Incident Linking Framework (ILF) is to build the semantic link between the tweet and incident database. This semantic link is established based on different features provided by tweets and incidents by checking the similarity between each tweet and the incident. Thus, it is not a binary decision but rather a similarity metric developed, which helps integrate the involved features with different scores.

The Incident Linking Framework contains three different steps necessary to link the tweets with the incident database. These steps are listed below and the pipeline of the ILF is shown in below Figure 3.1.

1. Pre-processing and classification for tweets
2. Candidate generation
3. Candidate ranking

As shown in Figure 3.1, pre-processing and classification component will take raw tweets as an input and will perform all the pre-processing steps. Then, we will give the resultant tweets and the incident database to the candidate generation component to generate the candidate sets. Each candidate in the candidate set contains the tuple (tweet, incident). Finally, the candidate ranking component will

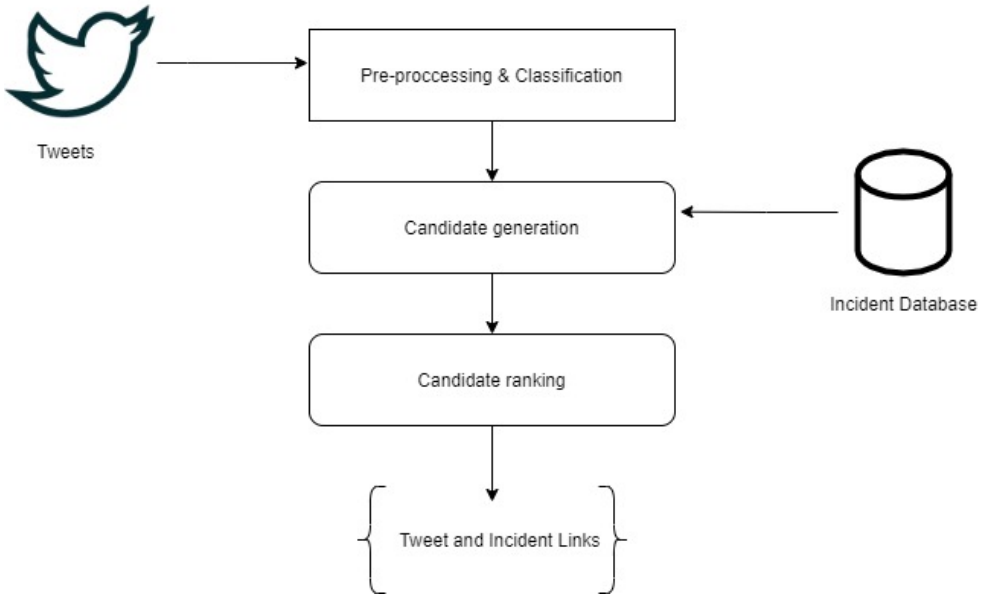

Figure 3.1: ILF pipeline which takes tweets and incidents as an input which gives tweets and incident links as an output

take the candidate set as an input, rank the individual candidate pair, and give top rank tweet and incident pairs.

3.1.1 Pre-processing and classification of tweets

Twitter users usually post their tweets with 280 characters short text. Unfortunately, these short texts are often hard to understand for a machine due to informal language, grammar, spelling errors and abbreviations and also contain irrelevant information like usernames, hashtags and URLs are not helpful to train a general classifier for various events. To avoid that a classifier learns these incident-specific patterns, we have to remove them from the text using text mining techniques. In our preprocessing step, we have removed all the unnecessary information present in the tweets listed below.

- URLs
- Hashtags, Mentions, Reserved words (RT, FAV)
- Emojis, Smileys
- Converted text into numbers if they present any

In the classification, we have obtained a pre-trained classification model from Wiegmann et al. (2020) to classify the tweets. This model is a state-of-the-art model that tells whether the tweet is related to a disaster. This step will allow us to remove the irrelevant tweets before giving it a model, and it improves the performance of the overall system.

3.1.2 Candidate generation

In this step, we primarily focus on extracting possible entities from the tweets called tweet entity mentions, including location, disaster type, impact (number of deaths), and time. We used pre-trained NER models like Spacy (Honnibal et al., 2020) to extract these possible entity mentions from the tweet. Consequently, we have extracted the corresponding incident entities from the incident databases. After extraction, we measure the similarity between an entity mention and the incident entity, and if it matches, we considered that incident entry as a candidate for that tweet. We have divided these candidates generation step into four sets based on the metadata that we can extract from the tweet. At the end of the candidate generation step, we have combined all these sets to generate one candidate set.

- Location-based candidates
- Disaster type-based candidates
- Impact-based candidates
- Time-based candidates

Location-based candidates

To generate the location candidate set, we have extracted the location mentions from the tweet using pre-trained models called Spacy (Honnibal et al., 2020) and nominatim (Sarah Hoffmann, 2020). After extraction, we retrieved location entities from the incident database and examined the coordinate differences and word similarity between extracted locations and locations presented in the incident database. If these two locations satisfy the following conditions, we added that as a candidate for that tweet.

1. If the location of the tweet matches the exact location of an incident location.
2. If tweet location or its country or its state corresponds with the precise location of an incident location.

We have implemented similarity metric $SimF_L(t_j^L, i_k^L)$ to check these. Apart from these conditions, we included the time constraint. If the difference between tweet time and incident time falls under a certain threshold, we only consider that as a candidate. We assign the threshold value based on the disaster type. The following formula was developed to generate the location-based candidate set C_L.

$$C_L = \{(t_j, i_k) \mid \forall\ j \in \{1, ..., n_t\}, k \in \{1, ..., n_i\}\ : SimF_L(t_j^L, i_k^L) \wedge \Delta T_{(t_j, i_k)} < \tau\}$$

where:

C_L = Location-based candidates

t = Tweet

i = Incident database entry

$SimF_L$ = Similarity function for location

t_j^L = Location mention for tweet j

i_k^L = Incident location entity

j, k = index values

$\Delta T_{(t_j, i_k)}$ = Difference between incident entry time and tweet time in hours

τ = time threshold (based on disaster type)

For example, NER recognizes "Columbia" in the tweet, and check for "Columbia" location across all entries of the database. If we find any, we consider that as a candidate for that tweet only if it falls under the given time threshold. Figure 3.2 illustrates the location mention in the tweet and the location entity in the incident database.

Disaster type based candidates

To generate the disaster type-based candidate set, we have built a separate list that contains related incident types and synonyms for each incident type. Then we extracted the disaster type mentions from the tweet based on the occurrence of one of the keywords in the list. We compared these extracted disaster type mentions with the corresponding entry in the incident database. If these two disaster types satisfy the following conditions, we only added that as a candidate for that tweet.

1. If the disaster type of the tweet matches the exact disaster type of an incident.
2. If the disaster type of the tweet and disaster type of an incident fits relatively (synonym of disaster type).

We have implemented similarity metric $SimF_D(t_j^D, i_k^D)$ to check these conditions. Apart from these, we included the time constraint. If the tweet time and incident

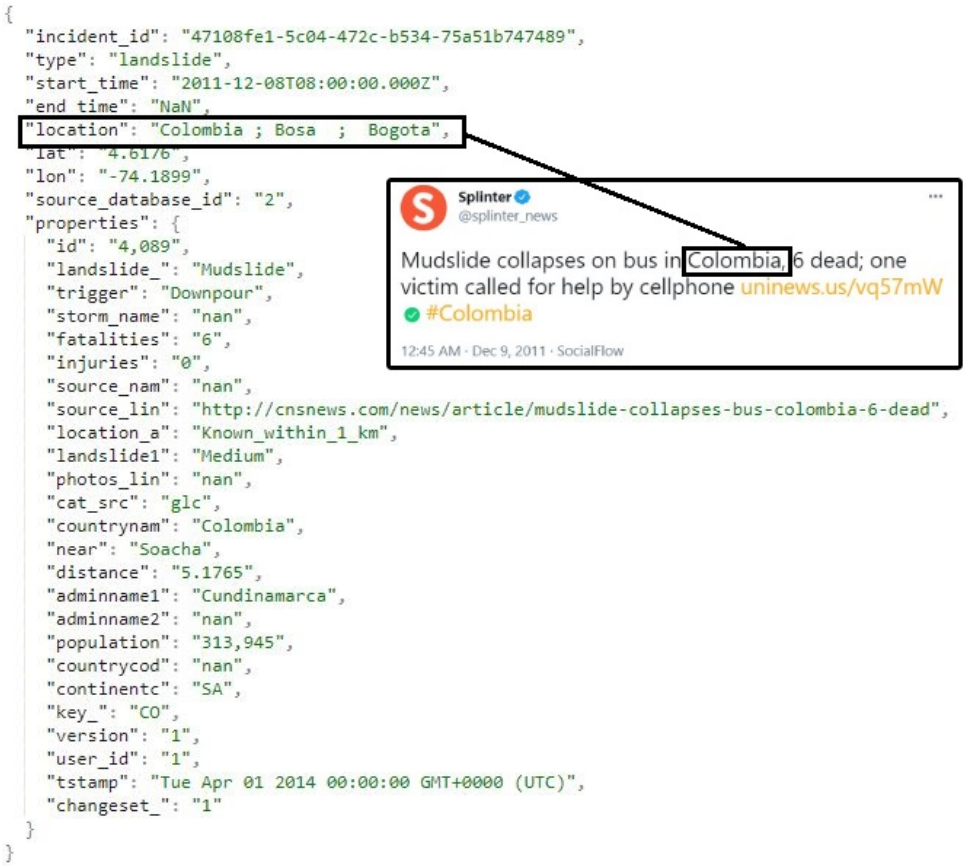

Figure 3.2: Tweet location - Incident location

time fall under a certain threshold, then only we consider that as a candidate. The following formula describes to generate the disaster type candidates.

$$C_D = \{(t_j, i_k) \mid \forall\, j \in \{1, ..., n_t\}, k \in \{1, ..., n_i\} \ : SimF_D(t_j^D, i_k^D) \wedge \Delta T_{(t_j, i_k)} < \tau\}$$

where:

C_D = Disaster type-based candidates

t = Tweet

i = Incident database entry

$SimF_D$ = Similarity function for disaster type

t_j^D = Disaster type mention for tweet j

i_k^D = Incident disaster type entity

j, k = index values

$\Delta T_{(t_j, i_k)}$ = Difference between incident entry time and tweet time in hours

τ = time threshold (based on disaster type)

For example, we found “mudslide” in the tweet, and we check for the mudslide or relative disaster type across the database. If we find any, we have added that as a candidate for that tweet and if it falls under the given time threshold. The following Figure 3.2 illustrates the disaster type mention in the tweet and the disaster type entity in the incident database.

Figure 3.3: Tweet disaster type - Incident disaster type

Impact based candidates

To generate the impact-based candidates, we extracted the cardinal mentions from tweets using NER. We compared these cardinal mentions with the corresponding number of deaths in the incident database. If these impact numbers satisfy the following conditions, we added that as a candidate for that tweet.

1. If the numerical value presented in the tweet matches the exact value of the number of deaths of an incident.

2. If the numerical value given in the tweet and number of deaths of an incident database fits relatively.

We have implemented similarity metric $SimF_I(t_j^I, i_k^I)$ to check these conditions. Apart from these, we included the time constraint. If the tweet time and incident time fall under a certain threshold, then only we consider that as the candidate. The following formula was used to generate the impact candidate set.

$$C_I = \{(t_j, i_k) \mid \forall\ j \in \{1, ..., n_t\}, k \in \{1, ..., n_i\}\ : SimF_I(t_j^I, i_k^I) \wedge \Delta T_{(t_j, i_k)} < \tau\}$$

where:

C_I = Impact-based candidates

t = Tweet

i = Incident database entry

$SimF_I$ = Similarity function for impact

t_j^I = Impact mention for tweet j

i_k^I = Impact disaster type entity

j, k = index values

$\Delta T_{(t_j, i_k)}$ = Difference between incident entry time and tweet time in hours

τ = time threshold (based on disaster type)

For example, If we found “6” in the tweet, we check for the fatalities or deaths across the incident database. If it exactly matches or partially matches (near to 6 like 7 or 5), we have added that as a candidate for that tweet and only if it falls under the given time threshold. The following Figure 3.2 illustrates the disaster type mention in the tweet and the disaster type entity in the incident database.

Figure 3.4: Tweet impact - Incident impact

Time based candidates

In this step, we have extracted the timestamp for each tweet and timestamp of incident entry in the disaster database. Based on these time stamps, we have calculated the difference between the tweet time and the starting time of the incident database entry. If the contrast of the time matches the threshold, we consider that tweet and incident as a candidate set. We assign the threshold value based on the disaster type like we mentioned above. The following formula was used to generate the candidate set for time.

$$C_T = \{(t_j, i_k) \mid \forall\ j = (1, ..., n_t), k = (1, ..., n_i) : \Delta T_{(t_j, i_k)} < \tau = True\}$$

where:

C_T = Time-based candidates

t = Tweet

i = Incident database entry

$\Delta T_{(t_j,i_k)}$ = Difference between incident entry time and tweet time (no of hours)

i, j = index values

τ = time threshold (based on disaster type)

The following Figure 3.5 illustrates the timestamp in the tweet and the disaster starting time entity in the incident database.

Figure 3.5: Tweet time - Incident timestamp

Candidate generation Algorithm

In previous sections, we have generated the individual candidate sets for each possible entity mentions extract from the tweet. Now we have combined all these separate candidate sets to make it one candidate set. The below algorithm works only for generating each candidate set that we defined in the above sections. Each time of generating the candidate set , the function $SimF_p(t_m, i_e)$ takes one of the parameters L (Location), D (Disaster-type), I (Impact) to check for the similarity. If it matches or partially matches, it returns the $True$. $\Delta(t_{time}, i_{time})$ is used only

to generate the time-based candidate set. If the contrast of the time matches the threshold, then the function returns the $True$. These returned individual candidate sets are combined to generate the final candidate set.

Algorithm 3.1: Candidate generation

Input: $Tweets\ T = t_n\ , n = 1, 2, \ldots, t_n$
$Incidents\ I = i_k\ , k = 1, 2, \ldots, k_n$
Output: $Individual\ candidate\ set\ C$
T_m = $ExtractMentions\,(T)$;
I_e = $GetEntities(I)$;
Let $C = \{\varnothing\}$
foreach *Tweet Features* $t_m \in T_m$ **do**
  foreach *Incident features* $i_e \in I_e$ **do**
    c = [$\varnothing$]
    if $time(t_m, i_e) < T$ *AND* $D == 1$ **then**
      $Flag$ = $SimF_p(t_m, i_e)$ *or* $\Delta(t_{time}, i_{time}) < \tau$; p = L,D,I
      if $Flag == True$ **then**
        $c.insert(t_m, i_e)$;
      end
    end
  end
  $C.insert(c)$
end
return C

- **ExtractMentions():** This function extracts the structured data from the unstructured tweet using pre-trained NER. It takes the tweets as an input, and it will identify the mentions in the tweet. For example, if we operate this algorithm for a location-based candidate set for tweet "Mudslide collapses on bus in Colombia, six dead" and this function extracts the location mention "Colombia" from the tweet.

- **GetEntities():** This function will get the required features from the incident database. It takes the incident database entries as an input, and it will extract the entities in the incident entries. For example, if we operate this algorithm for a location-based candidate set for incident database entry "Incidentid : 4718XXX , location : "Colombia", start_time : 2011-12-08T08:00:00.000Z , type : landslide" and this function returns the incident location entity "Colombia".

3.1.3 Candidate ranking

In the candidate set, many incidents are attached to a single tweet. We now have to decide which is the most probable incident. Our next task is to link each tweet with the most similar incident candidate and output the most likely candidate or none. To achieve this, we have developed a score metric for each candidate in the candidates list by measuring the similarity between tweet entity mention and incident entity. After adding all scores, we identified the top score candidate and established the semantic link between the tweet and the incident.

We have used four different scoring functions for each measure to generate individual scores. Individual scores are combined to get the total score for the candidate, and four scoring functions are described below.

Location score

To generate the location score, we have checked the similarity between the tweet location and incident location for the candidate in the candidate list. Then, we have tokenized the location in the disaster database into tokens. If tweet location and part of the incident location match, we will add a specific score. In the end, we will calculate the total score for location by summing all the scores and add them to the candidate set. We have implemented a $SimS_L$ score metric to calculate the specific score. This function will take tweets and incidents as input, and it will tokenize and calculate the individual similarity scores by checking how accurately they match. If it fits accurately, then the score is 0.5; if it matches partially, the score will be 0.25. At the end, these scores will be summed. A detailed example is also given below.

We have used the following function to check for the similarity score between the tweet location and incident location.

$$C_{LScore} = SimS_L(t, i), \forall (t, i) \in C_L$$

where:

C_{LScore} = Location similarity score

t, i = Tweet , Incident entry

C_L = Location-based candidate set

$SimS_L$ = Similarity score function for location

For example, let's take a tweet, which the location says “Colombia bosa,". On the other hand, we have two incident entries. Their location says "Colombia" and “Colombia; Bosa," respectively. We give these tweet and incident entries to the similarity score function $SimS_L$. The tweet and second incident entry combination returns more scores (0.5) than the first incident entry combination (0.25).

Disaster type score

To generate the disaster type score, we have estimated the similarity score between the mentioned disaster type in the tweet and the disaster type in the incident database. If the mentioned disaster type and incident disaster type matches accurately or matches its synonym, we add a specific score to the candidate. We have implemented a $SimS_D$ score metric to calculate the disaster type matching score. This function will take tweets and incidents as input, and it calculates the individual similarity scores by checking how accurately they match. If it fits accurately, then the score is 0.60; if it matches partially, the score will be 0.40. In the end, these scores will be summed. A detailed example is also given below.

We have used the following function to check for the similarity score between the tweet disaster type and incident disaster type.

$$C_{DScore} = SimS_D(t, i), \forall (t, i) \in C_D$$

where:

C_{DScore} = Disaster type similarity score

t, i = Tweet , Incident entry

C_D = Disaster type-based candidate set

$SimS_D$ = Similarity score function for disaster type

For example, let's take a tweet, which the disaster type says “landslide". On the other hand, we have two incident entries. Their disaster type says "landslide" and “mudslide," respectively. We give these tweet and incident entries to the similarity score function $SimS_D$. The tweet and first incident entry combination return more scores(0.6) than the second incident entry combination(0.4).

Impact score

To generate the disaster impact score, we evaluated the similarity score between the mentioned disaster impact (number) in tweet and the number of deaths in the incident database. If it matches exactly, then we add some score. An if it partially matches, it will add some score. In the end, we will combine all the scores and add them to the candidate. We have implemented a $SimS_I$ score metric to calculate the impact matching score. This function will take tweets and incidents as input, and it calculates the individual similarity scores by checking how accurately they match. The assigned scores are between 1 and 0. These scores are based on the accuracy of their match. A detailed example is given below.

We have written the following function to check for the similarity between the tweet impact and incident disaster impact.

$$C_{IScore} = SimS_I(t, i), \forall (t, i) \in C_I$$

where:

C_{IScore} = Impact similarity score

t, i = Tweet , Incident entry

C_I = Impact-based candidate set

$SimS_I$ = Similarity score function for impact

For example, tweet "Mudslide collapses on bus in Colombia, 6 dead" which is numerical value "6" in it. On the other hand, we have three incident database entries. These entries have recorded the no of deaths "6", "4" and "10" respectively. When we score these tweet with three incident database entries using $SimS_I$, the first combination of tweet and incident entry returns a higher score(0.5 - Exact match), and the second combination returns less score than the first combination (0.3 - partial match). Likewise, the third combination returns a lower score when compared to the second combination (0.2 - relatively partial matching).

Time score

We have evaluated the similarity score between the tweet time and incident starting time to generate the time score. We have added the score to the candidate by calculated based on the time difference between tweet time and the incident time. If time falls under certain threshold then we have added the score as "1" other wise we added as "0".

$$C_{TScore} = SimS_T(t, i), \forall (t, i) \in C_T$$

where:

C_{TScore} = Time similarity score

t, i = Tweet , Incident entry

C_T = Time-based candidate set

$SimS_T$ = Similarity score function for time

After generating these scores individually, we calculated the total score by enumerating all the scores and add these scores to the candidate list. We return the most likely candidate from this scored candidate list by considering the top score candidate in the candidate list.

Candidate Ranking algorithm

The following algorithm demonstrates the candidate ranking for the generated candidate set in the previous step.

1. **GetScore():** This function will consolidate all the individual scores that we determined previously. It will take tweet and incident as input, and it calculates the score between all the tweet entity mentions and incident entities separately. In the end, it will sum all the scores to generate the final score for a given combination of tweet and incident.

2. **GetTopRank():** This function will take the candidate with scores as an input and returns the candidate which has highest score.

Algorithm 3.2: Candidate ranking

Input: $Candidate(C)$
Output: L_{ie} $Linked\ candidates$
Let $C_L = \{\varnothing\}$
foreach *Tweet* t and $Incident$ $i \in C$ **do**
  l = $[\varnothing]$
  foreach i *in Incident* **do**
    $Totalscore$ = $GetScore(t, i)$
    $l.append\ (totalscore, i)$
  end
  if $length(l)$ == $True$ **then**
    $C_s.append(T, l)$;
  end
end
L_{ie} = $GetTopRank(C_s)$
return L_{ie}

Chapter 4

Experiments

This chapter's main goal is to evaluate our proposed method using different experiments with a given corpus. In Section 4.1, we overview datasets to conduct the experiments. In Section 4.2, we introduce the basic evaluation metrics for this study. In Section 4.3, we conducted two different experiments to evaluate our method.

4.1 Datasets

To evaluate our approach, we used a tweets datasets that consist of approximately 30% of the total Twitter stream in different sized time windows (one hour to one day) from December 2011 to October 2019. Thus, a total of 414 million tweets stand for the one-time span of a little less than 180 hours available. Automated filters reduced the number of tweets to 23,673 (Juch, 2021). Consequently, we have an incident database with the same time frame as tweets datasets.

Figure 4.1 shows the statistics of the 15 datasets and 15 incident databases. There is a total of 23,673 tweets and an average of 1,578 tweets in each dataset. There are 23,723 incident entries and an average of 1,581 entries in each dataset in the incident database.

Before we receive the datasets, manual annotations have been done on these datasets by Juch (2021). Manual annotations contain information about each tweet that could explicitly linked to an incident in a database accordingly. We use these annotations to verify our proposed approach. After examining these manual annotations, there are 15 datasets, among which eight datasets have the entries to match with the tweets. Therefore, we have considered only those datasets to apply to our method.

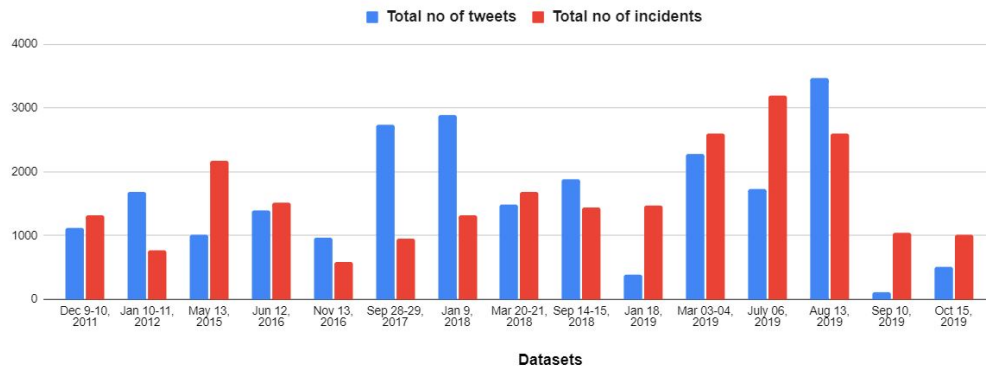

Figure 4.1: Complete statistics of the tweets count and incident database count

4.2 Evaluation metrics

To evaluate our method, we used the standard metrics for the linking-based system. These metrics include precision, recall, and F1 score. We have used the Mean Reciprocal Rank (MRR) for order in the search results. To achieve this, we have used the Mean Reciprocal Rank (MRR). To evaluate our proposed method, we have used F1- score to classification. To assess the candidate generation, we have used recall, and also we have applied MRR for candidate ranking.

In this thesis, we identify tuples (t,i) of one tweet t and one incident i, and the design of our method is intended to achieve high evaluation metrics. In the experiments, we investigate how well each step works and then determine which step should be adjusted. The following data table illustrates the confusion matrix for classification and candidate generation in ILF (Hand and Christen, 2018).

		True link status	
		Match	Non-match
Predicted link status	match	True positive (tp)	Flase positive (fp)
	non-match	False negative (fn)	True negative (tn)

Table 4.1: Data Table for evaluating ILF

1 Precision

The formula for the precision of ILF system is calculated using the below formula. The proportion of compared record pairs (Tweet, Incident) classified as matches that are true matches (Hand and Christen, 2018).

$$Precision = \frac{tp}{(fp + tp)}$$

2 Recall

The formula for the recall of ILF system is calculated using the below formula. The proportion of true matching record pairs (Tweet, Incident) that are classified as matches (Hand and Christen, 2018).

$$Recall = \frac{tp}{(fn + tp)}$$

3 F1-Score

In the above, we have seen the formulas for precision and recall. Sometimes these measures are consider together as F1- Score in order to calculate single evolution for entity system. The harmonic mean of precision and recall defined as a F1- Score (Wu et al., 2018).

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

4 Mean Reciprocal Rank (MRR)

Radev et al. (2002) defines the MRR as the multiplicative inverse of the rank of the first accurate answer. The mathematical formula for MRR is given below.

$$MRR = \frac{1}{\mid Q \mid} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

where $rank_i$ refers to the position of the first relevant document for the i-th query and we ignore the other relevant items if they present any. Entezari (2020) provided an example for MMR. In the Figure 4.2 , for the first query, the correct position is located at the 3rd position, and we assign 1/3 for that position. The scores are calculated for different queries, and we take the average scores for all queries.

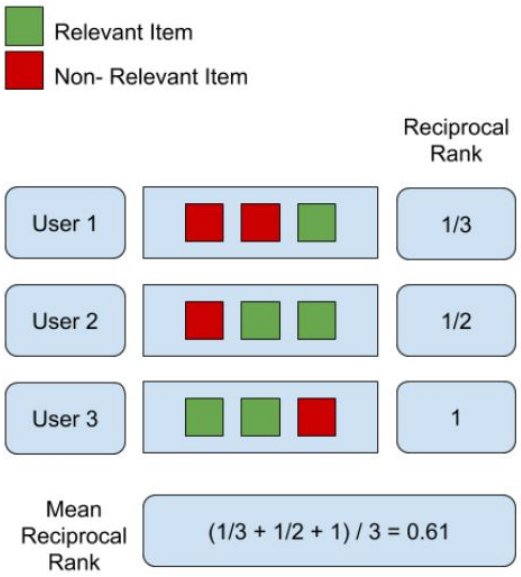

Figure 4.2: MRR Example Entezari (2020)

4.3 Experimental setup

To appropriately evaluate the contributions of this thesis, we carried out two experiments for the ILF method. For these two experiments, we somewhat adjusted the ILF approach. We named the original approach as ILF Method - 1 and the modified approach as ILF Method - 2. As discussed in Chapter 3, we have chosen four individual sets for candidate generation. In ILF Method - 1, we have selected only three candidate sets (location, disaster type, impact) in the candidate generation module, and we ignored the time candidate set. ILF Method - 2 chooses all the four candidate sets. The classification and ranking module will be the same for both methods and used before generating the candidates.

The main aim of these two experiments is to identify the importance of time constraints. We have checked the individual recall value for candidate generation and MRR for candidate ranking in every experiment. We have drawn the results by comparing both of these experiments. We have also evaluated the average number of candidate entries for each tweet in given datasets to check the system's performance. In the next chapter, we show the results of these mentioned experiments.

Chapter 5

Results and Discussion

This section outlines the experimental results and discussion of this study. As discussed in Chapter 4, we have considered two algorithms to evaluate our research. We have admitted only location, disaster type, and impact measures in the candidate generation module in the ILF Method - 1. In the ILF Method - 2 also follows the ILF Method - 1, but we also included the time criterion in it. The evaluation of these algorithms takes place in two measures called intrinsic metrics and extrinsic metrics.

- **Intrinsic metrics:** We evaluate each module individually without the side effects from others.
- **Extrinsic metrics:** we measure the whole application with cascading errors.

5.1 Classification

As discussed in Chapter 3, we have applied pre-trained models to our datasets, and we have calculated the F1- Score (Intrinsic). The following figure illustrates the individual f1-scores for each datasets. We have also calculated the micro average for this classification module, and it exhibits an F1- Score of 0.86.

Wiegmann et al. (2020) mentioned 0.88 is the best F1- Score for their experiments on tweet corpus 2020. In our case, we have got 0.86 which is very close to the best score. We have achieved good results by using these pre-trained models. Figure 5.2 reveals, that a lower F1-scoe of 0.64 was obtained for the dataset "Dec 9-10, 2011".

This pre-trained classifier which we used in the thesis, is purely language-dependent and will acknowledge the tweets only in English. From these results, we also observed a couple of tweets were identified as disaster-related tweets though no information helps to connect them with an incident.

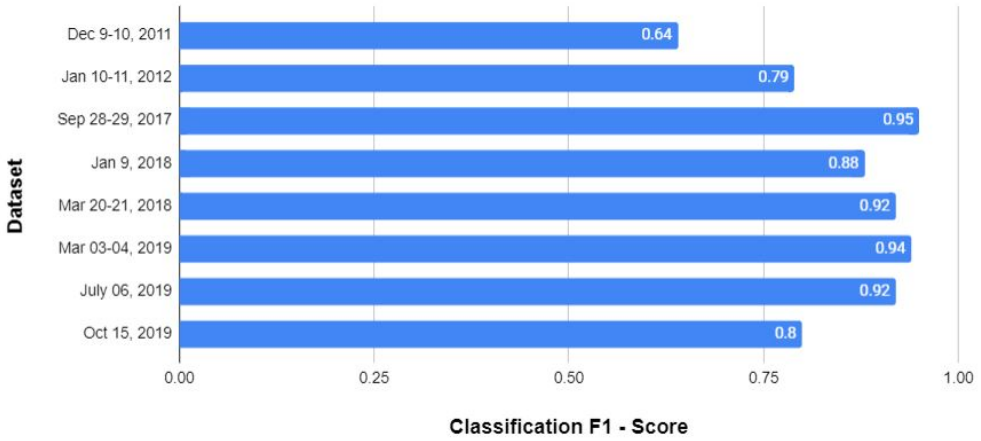

Figure 5.1: F1 - Scores of classification module (intrinsic)

5.2 Candidate generation

As we mentioned in Chapter 4, we have measured the recall value for both the methods, and the results are shown in the below figure. The micro average of ILF Method - 1 exhibits a recall value of 0.69, and ILF Method - 2 shows a better recall value of 0.89. We also calculated the count of the candidates for each algorithm. The average count of ILF Method - 1 is 95, and for ILF Method - 2, it's 418.37.

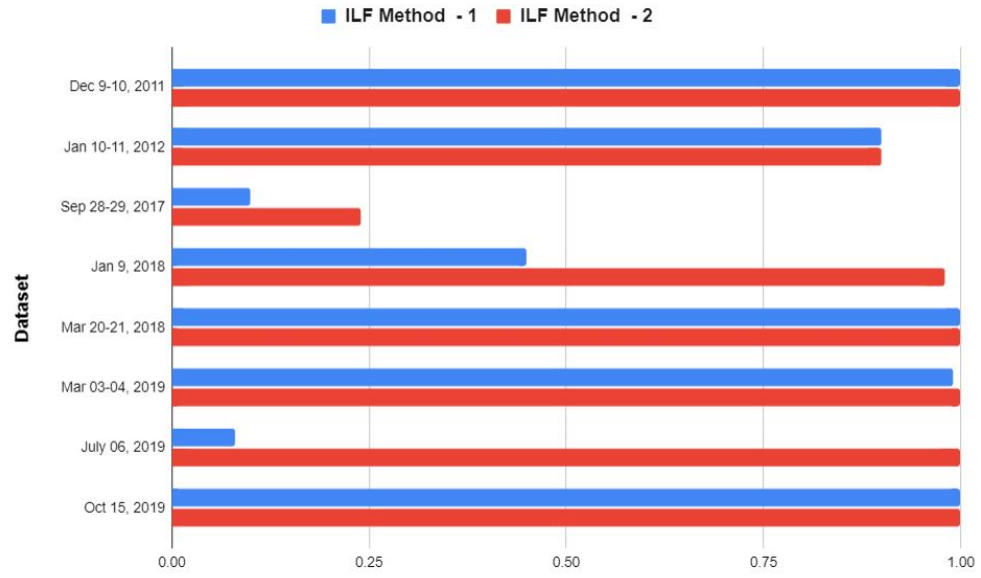

Figure 5.2: Candidate generation recall of datasets

As shown in the above figure, our candidate generation module exhibits good

results with both the algorithms except for few datasets like "Jan 9, 2018" and "July 06, 2019 ". For both datasets ILF Method - 2 performed better results than the ILF Method - 1. Though the recall was better for ILF Method - 2, it shows the poor performance because of the average count of each candidate set. The average count was more in ILF Method - 2 than ILF Method - 1.

In the candidate generation module, we initially used only spacy to recognize the location mentions in the tweet. The recall was very low because Spacy could not identify the few locations present in the tweet (Eg. Frisco). Later, we used two different NER's (Spacy and Nominatim) to make the better recall.

5.3 Candidate ranking

We have measured the individual candidate ranking module with intrinsic measures, and we use extrinsic measures to evaluate the overall system with cascading errors. The best score for MRR is 1, and the lower score is 0. Below, Figure 5.3 shows the comparison of MRR for ILF Method - 1 and ILF Method - 2. The average MRR (Intrinsic) of ILF Method - 1 and 2 is 0.1912 and 0.0329, respectively.

As manifested in the below Figure 5.3, ILF Method - 1 shows better MRR values than the MRR values of ILF Method - 2, but for the datasets like "Mar 20-21, 2018" and "Mar 03-04, 2019" demonstrated poor results. Nevertheless, the system performance was also good for ILF Method - 1 when compare to ILF Method - 2.

The ILF Method - 1 worked so well on the Oct 15, 2019 dataset because the dataset was small, and the average number of candidates for each tweet is 52 compared to other datasets. Two data sets return MRR value as 0. Though these datasets performed well in the candidate generation, they did not work well in the candidate ranking. For better MRR we need to adjust the scoring functions.

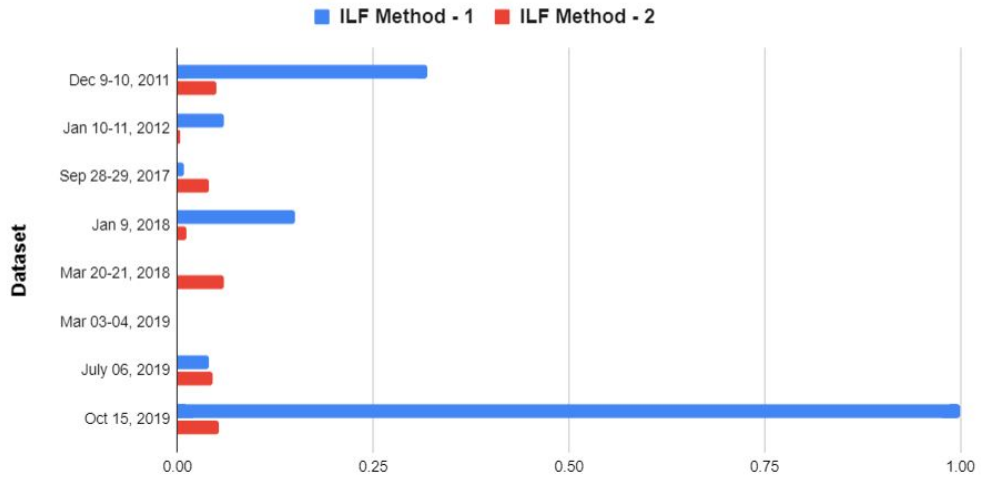

Figure 5.3: MRR values of ILF Method - 1 Vs ILF Method - 2 (Intrinsic)

Figure 5.4 shows the extrinsic values for MRR of ILF Method - 1 and 2. The average MRR (Extrinsic) of ILF Method - 1 and 2 is 0.1912 and 0.0329, respectively.

As illustrated in the above figure, the overall MRR values of ILF Method - 1 expose better results than ILF Method - 2. However, even if ILF Method - 1 shows better results for half of the datasets, it shows poor results. Therefore, the overall performance of algorithm1 shows better results than ILF Method - 2.

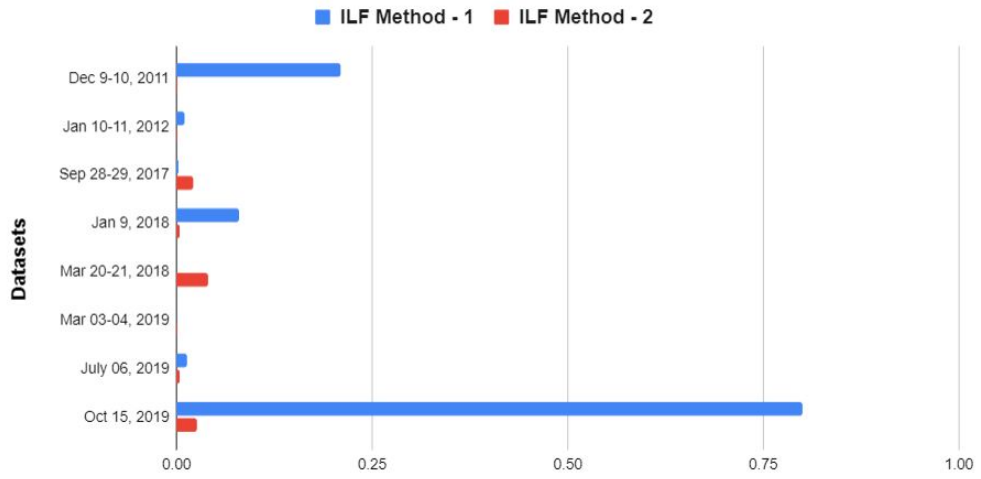

Figure 5.4: MRR values of ILF Method - 1 Vs ILF Method - 2 (Extrinsic)

Table 5.1 and Table 5.2 shows the overview results for all the datasets for ILF Method - 1 and ILF Method - 2. In addition, we have also calculated the micro averages for all the algorithms shown in Table 5.3.

Datasets	Classification F1 Score (Intrinsic)	Candidate generation recall (Intrinsic)	Candidate generation count (Intrinsic)	Candidate ranking MRR (Intrinsic)	Candidate ranking MRR (Extrinsic)
Dec 9-10, 2011	0.64	1	61.62	0.3200	0.2100
Jan 10-11, 2012	0.79	0.9	47.32	0.0600	0.0100
Sep 28-29, 2017	0.95	0.1	129.03	0.0080	0.0030
Jan 9, 2018	0.88	0.45	94.56	0.1500	0.0800
Mar 20-21, 2018	0.92	1	158.63	0	0
Mar 03-04, 2019	0.94	0.99	128	0	0
July 06, 2019	0.92	0.08	108	0.0400	0.0135
Oct 15, 2019	0.8	1	52	1	0.8000

Table 5.1: Overview results of ILF Method - 1

Datasets	Classification F1 Score (Intrinsic)	Candidate generation recall (Intrinsic)	Candidate generation count (Intrinsic)	Candidate ranking MRR (Intrinsic)	Candidate ranking MRR (Extrinsic)
Dec 9-10, 2011	0.64	1	977	0.0500	0.0010
Jan 10-11, 2012	0.79	0.90	462	0.0031	0.0015
Sep 28-29, 2017	0.95	0.24	557	0.0400	0.0215
Jan 9, 2018	0.88	0.98	918	0.0116	0.0044
Mar 20-21, 2018	0.92	1	145	0.0600	0.0400
Mar 03-04, 2019	0.94	1	128	0.0010	0.0004
July 06, 2019	0.92	1	108	0.0452	0.0034
Oct 15, 2019	0.8	1	52	0.0530	0.0265

Table 5.2: Overview results of ILF Method - 2

Datasets	Classification F1 Score (Intrinsic)	Candidate generation recall (Intrinsic)	Candidate generation count (Intrinsic)	Candidate ranking MRR (Intrinsic)	Candidate ranking MRR (Extrinsic)
ILF Method - 1	0.84	0.69	95	0.1972	0.1395
ILF Method - 2	0.84	0.89	418	0.0329	0.0123

Table 5.3: Overview (micro average)

Overall, the candidate generation module worked well in terms of recall. The candidate ranking module is obtained a lower MRR for most datasets because the average number of candidates is so high for each tweet. Improvement in the similarity function that is used in candidate generation will reduce the no of candidates. In ILF Method - 1, the candidate ranking module failed in two datasets. It means that there is no matching entry related to any of the tweets in those datasets. By adjusting the scoring functions that we discussed in Chapter 3, it may improve the better results.

Conclusively, we have answered the RQ1 in chapter 3 to extract the different features like location, disaster type, impact, and time from the tweets that match the entries in the incident database (Knowledge database). To answer RQ2, we have used the features mentioned above to build the ILF that identifies all the tweets and matching incident database entries (explained in chapter 3). In this chapter, we have answered RQ3 about the efficiency of the proposed approach.

Chapter 6

Conclusion

This thesis explored a new methodology for linking each tweet with the incident database. We have also investigated and developed two key components of the incident linking framework, candidate generation and candidate ranking, and used the pre-trained model for classification. Based on the features that we can extract from the tweet.

Moreover, we have developed two algorithms for candidate generation. We verified one of its candidate generation modules was so influential on our datasets. From our experiments conducted in Chapter 4, we assumed that better location identifiers (NER) make our candidate generation algorithms more powerful. Furthermore, we have also learned that using two different NER's makes better recall. Still, the system's performance will get lower due to the heavy no of candidates generated by the system.

Finally, we can use this method to link the tweets with the incident database, but candidate generation and candidate ranking modules need to be improved for better results. For example, in our candidate generation module, the average number of incidents for each tweet is more than the usual number (30-40). These numbers affect the candidate ranking module with MRR.

6.1 Future Work

Several things may be worth some additional research and this thesis serves as reasonable beginning points for any future study.

In our approach, we have mainly concentrated on linking the tweets with the incident database entries. Still, we may be enhancing this system by creating the missing entries in the database.

Location extraction from tweets is a critical aspect (e.g., disambiguation (state, country, or city-level? which country?) or realizing that there was a mention of a Mountain) that may drastically enhance the performance.

Our proposed approach is confined to English-related tweets. Further research on this study, we plan to use the technology of cross-lingual information linking of the tweets. We have employed only the probabilistic record linkage method for candidate ranking, where we add some weight based on the similarity. We may be leveraging advanced techniques in future work, such as Convolutional Neural networks (CNN) and Long Short Term Memory (LTSM) networks, to rank the candidates.

Bibliography

Heike Adel. Deep learning methods for knowledge base population, June 2018. URL http://nbn-resolving.de/urn:nbn:de:bvb:19-228945. 2.3

Majid Asgari-Bidhendi, Behrooz Janfada, and Behrouz Minaei-Bidgoli. Farsbase-kbp: A knowledge base population system for the persian knowledge graph. *Journal of Web Semantics*, 68:100638, 2021. ISSN 1570-8268. doi: https://doi.org/10.1016/j.websem.2021.100638. URL https://www.sciencedirect.com/science/article/pii/S1570826821000135. 2.3

Drake Baer. As sandy became sandy, emergency services got social, Nov 2012. URL https://www.fastcompany.com/3002837/sandy-became-sandy-emergency-services-got-social. 1

Z. Chen, Suzanne R. Tamang, Adam Lee, Xiang Li, W. Lin, Matthew G. Snover, J. Artiles, Marissa Passantino, and Heng Ji. Cuny-blender tac-kbp2010 entity linking and slot filling system description. *Theory and Applications of Categories*, 2010. 2.1.1, 2.1.1, 2.1.2

Zheng Chen and Heng Ji. Collaborative ranking: A case study on entity linking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 771–781, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D11-1071. 2.1.2

Silviu Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D07-1074. 2.1.3

S. Deorowicz and M. Ciura. Correcting spelling errors by modelling their causes.

International Journal of Applied Mathematics and Computer Science, 15:275–285, 2005. 2.1.1

S. Dusetzina, S. Tyree, Adrian Meyer, A. M. Meyer, Laura Green, and W. Carpenter. Linking data for health services research: A framework and instructional guide. 2014. 2.2.2

Mohamed Elfeky, Vassilios Verykios, Ahmed Elmagarmid, Thanaa Ghanem, and Ahmed Huwait. Record linkage: A machine learning approach, a toolbox, and a digital government web service. 04 2003. 2.2.3

Saeed Entezari. Deep Neural Ranking Models for Argument Retrieval. Master's thesis, Bauhaus-Universität Weimar, Fakultät Medien, Computer Science and Media, September 2020. (document), **4**, 4.2

Zheng Fang, Yanan Cao, Ren Li, Zhenyu Zhang, Yanbing Liu, and Shi Wang. High quality candidate generation and sequential graph attention network for entity linking. In *Proceedings of The Web Conference 2020*, WWW '20, page 640â650, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380146. URL `https://doi.org/10.1145/3366423.3380146`. 2.1.1

Jeremy Getman, Joe Ellis, Stephanie Strassel, Zhiyi Song, and Jennifer Tracey. Laying the groundwork for knowledge base population: Nine years of linguistic resources for TAC KBP. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL `https://www.aclweb.org/anthology/L18-1245`. 2.3

Michael Glass and Alfio Gliozzo. A dataset for web-scale knowledge base population. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 256–271, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93417-4. (document), 2.3, 2.3, 2.4, 2.5, 2.3

Thomas Gschwind, Christoph Miksovic, Julian Minder, Katsiaryna Mirylenka, and Paolo Scotton. Fast record linkage for company entities, 2019. 2.2.3

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194:130–150, 2013. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2012.04.005. URL `https://www.sciencedirect.com/science/article/pii/S0004370212000446`. Artificial Intelligence, Wikipedia and Semi-Structured Resources. 2.1.1

Gareth Hagger-Johnson, Katie Harron, Tom Fleming, Ruth Gilbert, Harvey Goldstein, Rebecca Landy, and Roger C Parslow. Data linkage errors in hospital administrative data when applying a pseudonymisation algorithm to paediatric intensive care records. *BMJ Open*, 5(8), 2015. ISSN 2044-6055. doi: 10.1136/bmjopen-2015-008118. URL https://bmjopen.bmj.com/content/5/8/e008118. 2.2.1

Xianpei Han and Jun Zhao. $\text{Nlpr}_k bpintac2009kbptrack : Atwo - stagemethodtoentitylinking$. *Theory and Applications of Categories*, 2009. 2.1.1, 2.1.3

David Hand and Peter Christen. A note on using the f-measure for evaluating record linkage algorithms. 28(3), 2018. ISSN 0960. 4.2, **1**, **2**

Katie Harron, Chris Dibben, James Boyd, Anders Hjern, Mahmoud Azimaee, Mauricio L Barreto, and Harvey Goldstein. Challenges in administrative data linkage for research. *Big Data & Society*, 4(2):2053951717745678, 2017. doi: 10.1177/2053951717745678. URL https://doi.org/10.1177/2053951717745678. PMID: 30381794. 2.2.2

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303. 3.1.2, 3.1.2

Felix Juch. Verknuepfen von tweets mit wissensdatenbanken zur analyse von gefahrenereignissen, Mar 2021. 4.1

Md. Yasin Kabir and Sanjay Madria. A deep learning approach for tweet classification and rescue scheduling for effective disaster management, Aug 2019. URL https://arxiv.org/abs/1908.01456. 1

D. Krewski, Y. Wang, S. Bartlett, Jan Zielinski, and R. Mallick. The effect of record linkage errors on statistical inference in cohort mortality studies. 01 2001. 2.2.2

John Lehmann, Sean Monahan, Luke Nezda, A. Jung, and Ying Shi. Lcc approaches to knowledge base population at tac 2010. *Theory and Applications of Categories*, 2010. 2.1.1

Rayees Ibrahim Lone and Dr S. Subramani. Natural Disasters: Causes, Consequences and Its Preventive Role in Sustainable Development. *International Journal of Indian Psychology*, 3(3), June 2016. ISSN 2348-5396. doi: 10.25215/0303.066. URL https://ijip.in/articles/natural-disasters-causes-consequences-and-its-preventive-role-in-sustainable-development/. 1

NCEI. Storm events database. URL https://www.ncdc.noaa.gov/stormevents/. 1

Anja Pilz and Gerhard Paaß. From names to entities using thematic context distance. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 857â866, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450307178. doi: 10.1145/2063576.2063700. URL https://doi.org/10.1145/2063576.2063700. 2.1.2

Abhaya S. Prasad and Louis Hugo Francescutti. Natural disasters. In Stella R. Quah, editor, *International Encyclopedia of Public Health (Second Edition)*, pages 215–222. Academic Press, Oxford, second edition edition, 2017. ISBN 978-0-12-803708-9. doi: https://doi.org/10.1016/B978-0-12-803678-5.00519-1. URL https://www.sciencedirect.com/science/article/pii/B9780128036785005191. 1

Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. Evaluating web-based question answering systems. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain, May 2002. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2002/pdf/301.pdf. 4

Henry Rosales Mendez, Barbara Poblete, and Aidan Hogan. What should entity linking link? 05 2018. 2.1

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613â620, November 1975. ISSN 0001-0782. doi: 10.1145/361219.361220. URL https://doi.org/10.1145/361219.361220. 2.1.2

Brian Quinion Sarah Hoffmann, Mtmail, 2020. URL https://nominatim.org/release-docs/2.5.1/. 3.1.2

Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2015. doi: 10.1109/TKDE.2014.2327028. (document), 2.1, 2.1, 2.1.1, 2.1.1, 2.1.1, 2.1.1, 2.1.1, 2.1.2

USGS. U.s. geological survey. URL https://www.usgs.gov/. 1

Vasudeva Varma, P. Pingali, Rahul Katragadda, Sai Krishna, S. Veeravalli, Kiran Sarvabhotla, Harish Garapati, Hareen Gopisetty, V. B. Reddy, B. KranthiReddy, Praveen Bysani, and R. Bharadwaj. Iiit hyderabad at tac 2009. *Theory and Applications of Categories*, 2008. 2.1.1

Margot Whitney. 40 twitter statistics marketers need to know in 2020, 2020. URL https://www.wordstream.com/blog/ws/2020/04/14/twitter-statistics. 1

Emily R. Wiegand and Robert M. Goerge. Record linkage innovations for the human services. *Record linkage innovations for the human services. Washington*, 2019. URL https://www.chapinhall.org/wp-content/uploads/PDF/Record-Linkage-Innovations-for-the-Human-Services.pdf. 2.2, 2.2.1, 2.2.2

M. Wiegmann, J. Kersten, H. Senaratne, M. Potthast, F. Klan, and B. Stein. Opportunities and risks of disaster data from social media: a systematic review of incident information. *Natural Hazards and Earth System Sciences*, 21(5):1431–1444, 2021. doi: 10.5194/nhess-21-1431-2021. URL https://nhess.copernicus.org/articles/21/1431/2021/. 1

Matti Wiegmann, J. Kersten, Friederike Klan, Martin Potthast, and Benno Stein. Analysis of detection models for disaster-related tweets. 2020. 3.1.1, 5.1

Gongqing Wu, Y. He, and Xuegang Hu. Entity linking: An issue to extract corresponding entity with knowledge base. *IEEE Access*, 6:6220–6231, 2018. 2.1.2, 2.1.2, **3**

Wei Zhang, C. Tan, Yan Chuan Sim, and J. Su. Nus-i2r: Learning a combined system for entity linking. *Theory and Applications of Categories*, 2010. 2.1.2

Wei Zhang, Yanchuan Sim, Su Jian, and Chew Lim Tan. Entity linking with effective acronym expansion, instance selection and topic modeling. pages 1909–1914, 01 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-319. 2.1.1