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

Prathyusha Nerella
ESRec, Bern University of Applied Sciences
Biel, Switzerland
prathyusha.nerella@bfh.ch

Akansha Bhardwaj, Paolo Rosso, Philippe Cudré-Mauroux
eXascale Infolab, University of Fribourg
Fribourg, Switzerland
{akansha.bhardwaj, paolo.rosso, pcm}@unifr.ch

***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 organizations. 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.**

***Index Terms*—Entity Linking, Natural Language Processing, Knowledge Base, Entity Disambiguation, Microposts, Twitter**

I. Introduction

Microblogging services allow users to post personal messages in real-time. For example, Twitter[1] is one of the widely used microblogging platforms where users post messages. These messages, commonly referred to as tweets, form an important source of instant information on any topic, including celebrity gossip, entertainment, news, etc. Consequently, the development of specific techniques for analyzing tweets has attracted considerable attention in recent years [1]–[4].

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 in which extracted mentions from text are mapped onto existing entities in a reference Knowledge Base. Some of the widely used Knowledge Bases in this context are Wikidata[2], DBpedia[3], and Yago 4 [5]. In this work, we focus on Entity Disambiguation on tweets, with Wikidata as our reference Knowledge Base.

Table Ia 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 Ib, each mention in a tweet typically can be mapped onto multiple candidate entities.

There are two specific key challenges that make it hard to disambiguate entities in tweets. First, tweets are short texts containing little information about the context. Second, tweets are informal in their style making their interpretation difficult. 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:

- We study how clustering tweets semantically can help compensate for the lack of context, and how semantic clusters can be leveraged to improve the Entity Disambiguation process.
- We propose a novel two-step process for Entity Disambiguation in tweets. Our proposed approach outperforms the state of the art on several gold standard datasets by up to 15.13%.
- We make our implementation code and datasets fully available for reproducibility purposes[4].

The rest of this paper is structured as follows. Section 2 presents related work on Entity Disambiguation. In Section 3, we describe our approach towards Entity Disambiguation. Section 4 presents our experimental setup, as well as the datasets we use in our empirical evaluation and our results. In Section 5, finally, we conclude our work, present some of its limitations, and discuss potential future work.

II. Related work

We start by introducing a few terms that will be used throughout our paper. An *entity* represents an instance present in a Knowledge Base. A *mention* is a word or a word phrase occurring in the text that has the potential to be mapped onto an entity in the Knowledge Base. For each *mention*, an Entity Disambiguation algorithm considers one or more entities in the Knowledge Base as potential candidates; these entities are called *candidate entities*. A disambiguation algorithm maps the

[1]https://twitter.com/
[2]https://www.wikidata.org/
[3]http://www.dbpedia.org/
[4]https://github.com/eXascaleInfolab/tweets_annotation

Example Tweet
Western envoys tout deal on core of U.N. **Syria** draft; **Russia** denies: **UNITED NATIONS** (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
Russia	Q34266, Q42195226, Q159, Q7381938, Q24058892	Q159
Syria	Q3509007, Q207118, Q21441670, Q36882108, Q858	Q858
UNITED NATIONS	Q7888316, Q7888319, Q1065, Q7888314, Q2943350, Q7888317	Q1065
US	Q53552040, Q30, Q1456659, Q28775762, Q11252141	Q30

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

TABLE I: (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.

mention onto its correct *entity* counterpart from the Knowledge Base.

Entity extraction is the task that deals with identifying all valid mentions. This task is also referred to as Mention Detection. Entity Linking, or Entity disambiguation is the task that maps the detected mentions to the correct Entity in the Knowledge Base. Such approaches have been applied on long documents like web pages and also on informal, shorter pieces of text like tweets and news articles.

A. Entity Linking on Documents and Tables

AIDA [6] is an online tool for entity disambiguation for text and tables using the YAGO2 Knowledge Base [7]. AIDA uses Stanford NER [8] to identify the mentions but also takes into account manually provided mentions from users. DoSeR [9] introduced a global disambiguation approach for Web table entity disambiguation. The algorithm trains a Word2vec [10] model for all entities and uses the resulting model for calculating the similarity between entities. This approach was also used for entity disambiguation in tables [11].

Eslahi et al. [12] optimized the above approach [9] by introducing an iterative model to perform the Web table annotation task. Their proposed iterative method assigns more weight to the mentions with unique candidates and builds an initial graph leveraging these unique candidates. Then at every iteration, the algorithm considers one ambiguous mention to disambiguate, and adds it to the initial graph at the end of the iteration. Our approach also takes an iterative approach towards disambiguation, but as we focus on microposts, our methodology to build a disambiguation graph is different (more details on this point in Section III).

B. Entity Linking on Short, Informal Text

WAT-API [13] is an online Entity Linking tool and the successor of TagMe [14], a popular disambiguation tool. WAT-API annotation algorithm consists of three steps: Spotting, Disambiguation, and Pruning. 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.

Entity Linking of tweets based on dominant entity candidates (ELTDS) [15] is another closely related work. ELTDS works on the hypothesis that few 'dominant' entity candidates can be used for entity disambiguation of the tweets instead of all the available candidates.

III. Methodology

This section presents GraphEDM, our Entity Disambiguation framework for microposts. Figure 1 summarizes the method we adopt to disambiguate microposts. Our approach is divided into two phases: Context extension and Entity Disambiguation.

A. 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 of a tweet. We extend the context by first finding related tweets and then borrowing context from them. This process takes place in three steps. First, we preprocess the textual contents of the tweets. Next, we convert each tweet into a TF-IDF [16] or embedding representation [10]. Finally, these vector representations are used to generate clusters of related tweets. We explain the process in detail below:

1) ***Preprocessing****:* The text of a tweet is limited to 280 characters, and the content is often informal with slang, abbreviations, emoticons, URLs, and Hashtags [17], [18]. Thus, preprocessing is a necessary first step before tweet analysis. In this step, we remove emoticons, punctuation, web URLs, and HTML references in the tweets as this information is not essential for our task. Next, we tokenize the tweet text by converting each sentence into individual informative tokens by removing stop words and applying stemming techniques.

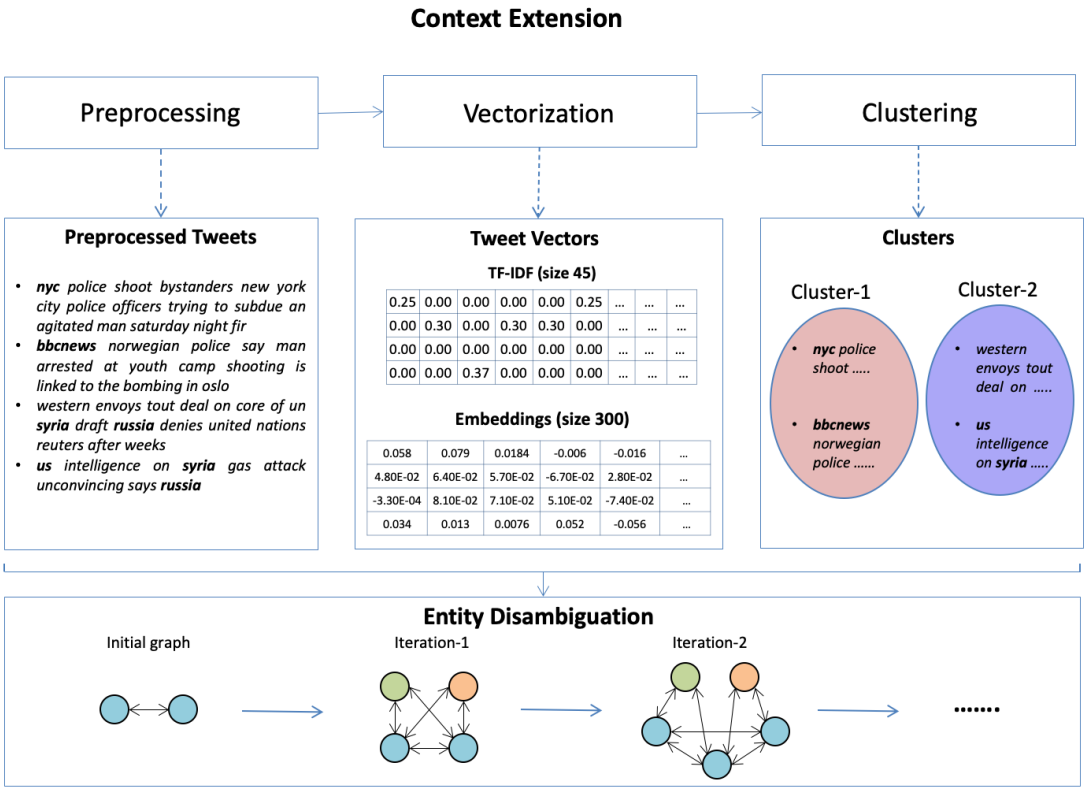

Fig. 1: The complete methodology of our proposed approach, GraphEDM. The approach is divided into two phases, Context Extension, followed by Entity Disambiguation. The words in bold are mentions in a tweet.

*2) **Vectorization**:* Vectorization is the process of converting a string into a unique numerical type, essential for many machine learning algorithms. For our clustering process, tweets are first converted into vectors using embeddings and TF-IDF representations.

We consider each tweet in the dataset as a document and calculate TF-IDF values for each word appearing in a tweet for vectorization. Hence, each tweet in the dataset has a dimension equivalent to the number of words in the dataset.

For the embedding representation, we used the publicly available pre-trained Google Word2vec model[5]. Each tweet is represented by an average of the vectors of the words present in that tweet.

*3) **Clustering**:* The final step in the process of generating a more meaningful context for a tweet is clustering. We use vectors from the previous step as input to our unsupervised clustering algorithms. Several clustering techniques have been proven to be effective in this context. In this work, we use K-Means [19], K-Medoids [20], Hierarchical Agglomerative (HAC) [21], Affinity Propagation [22], Hybrid Hierarchical K-Means [23], and Louvain [24] clustering techniques 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 a set of clusters of semantically related tweets.

In addition, we use the Louvain clustering [24] method, where 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.

B. Entity Disambiguation

The next step after clustering semantically related tweets is Entity Disambiguation. 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 1) in the following way:

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

1) First, we fetch all the mentions from the tweets within a cluster. For each mention, we fetch candidate entities by querying a surface form index[6].
2) We build an initial graph using unambiguous mentions (obtained in the above step) where all nodes are connected to each other.
3) Given an ambiguous mention m, we construct a k-partite graph for all k candidates of the mention m.
4) We resolve the mention m, and add it to our initial graph.
5) This process is repeated for each ambiguous mention of each cluster iteratively.

Describing Step 4 in more detail, we now explain the process of resolving an ambiguous mention m. First, we train a Word2vec model using all the candidate entities from the Wikidata Knowledge Base. We build a Word2vec training dataset 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, we represent each node 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 1:

$$weight(v_1, v_2) = \frac{cos(emb(v_1), emb(v_2))}{\sum_k cos(emb(v_1), emb(k))} \quad (1)$$

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

The PageRank [25] centrality ranking method is applied to the resulting 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). We fetch the type of the mentions (property: P31) and select the three major types existing among all the mentions. 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 we do not find any unambiguous mentions to generate the initial graph, the ambiguous mention in the first iteration is also disambiguated using 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 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.

[6]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*.

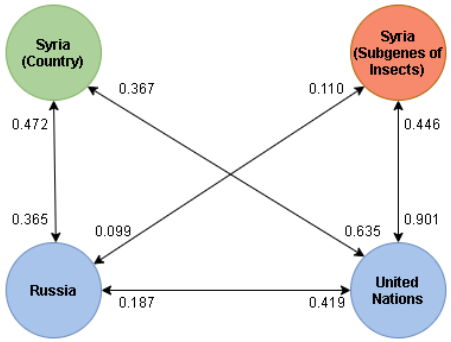

Fig. 2: Entity disambiguation with our iterative method

IV. 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.

A. Datasets

Table II gives key information about the seven publicly available datasets 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 II: Dataset statistics

Dataset	#Tweets	#Unique Entities	#Total Entities
Micropost2014 Test [26]	698	617	1014
Micropost2014 Training [26]	1518	1522	2938
Micropost2016 Test [27]	296	195	737
Micropost2016 Training [27]	4073	2789	6368
Micropost2016 Development [27]	100	98	253
Brian Collection [28]	1603	384	1231
Mena Collection [29]	162	348	482

B. Experimental Setup

In this subsection, we introduce our experimental setup, the baselines against which we compare, followed by the research questions that we try to answer.

We use Wikidata Knowledge Base for entity disambiguation. Levenshtein distance is used for those mentions that do not have a matching surface form in the Wikidata Knowledge Base when querying the mention against Wikidata KB.

*1) **Baselines**:* We compare the performance of our entity disambiguation framework against three state-of-the-art baselines – AIDA [6], WAT-API [13], and ELTDS [15]. We use standard precision, recall, and F1 scores as evaluation metrics for all methods.

C. 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 performance?

We answer each of those questions below.

D. Results - Q1

Table III shows the results of our proposed framework GraphEDM compared against state-of-the-art baselines for Entity Disambiguation. We observe that our proposed method outperforms all baseline methods on five datasets and are very competitive on the other two datasets. We choose the K-Means clustering technique and Word2vec embeddings for vectorization, for the results shown in Table III. 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 IV-E, IV-F).

We notice that the precision of WAT-API is higher 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. In contrast, our method results in a high recall compared to other methods as our candidate generation method that uses a surface form index can retrieve more relevant entities.

We give below a short analysis 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 with respect to F1 score. 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 about 3% less than 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 III: 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

E. Results - Q2

We experimented with multiple clustering techniques for the Context Extension stage of our framework. The clustering techniques used in this stage are K-Means, K-Medoids, Hierarchical Agglomerative Clustering (HAC), Affinity Propagation, Hybrid Hierarchical K-Means (HHK-Means), and the Louvain Clustering. For the Micropost 2014 Training dataset, Louvain shows the highest performance, while Agglomerative clustering has the lowest performance, which is 3.2% lower than Louvain. For the Micropost 2014 Test dataset, the performance with all the clustering techniques is almost similar, with the highest and the lowest scores differing only by 1%. For the Micropost 2016 Training dataset, our system performed better with the HHK-Means clustering with an F1 score of 57.8% while the performance is the lowest with the Louvain method with an F1 score 55.8%. For the Micropost 2016 Test dataset, our system reaches the best score of 61.9% and while the lowest score of 59.7% is obtained by the Agglomerative and Affinity clustering methods. For the Micropost 2016 Development dataset, the F1 score of our system with the various clustering techniques ranges between 57.7% and 63.2%. The highest and the lowest scores are obtained with Affinity Propagation and HHK-Means, respectively. For the Brian Collection, our system reaches an F1 score of 64.1% with K-Means clustering and 51% with Agglomerative clustering, which are the highest and lowest scores. The performance of our system on this dataset is spread across a wide range of values. For the Mena collection, the F1 scores for all the clustering techniques range between 76.3% and 73.9%. K-Means and Affinity propagation both have the same best score. The lowest score results from the HHK-Means clustering method.

We observe that performance was mainly dependent on

the dataset at hand. Our hybrid clustering technique, that combines Agglomerative and K-Means clustering, performed best overall and reached top scores on two different datasets, Micropost2014 Test and Micropost2016 Training.

F. 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. 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. On average, our results are better with Embeddings by up to 1.6% with respect to F1-score. In large datasets, TF-IDF vectors can be sparse, which leads to influencing the context extension phase negatively. On the other hand, embeddings are trained on a large datasets and are very effective at representing context.

V. Conclusion and Future work

In this work, we proposed GraphEDM, an effective framework for entity disambiguation in microposts. Our proposed method is composed of two phases. To extend the limited context given by the tweets, we first use clustering techniques to regroup semantically similar messages. Then, we disambiguate the mentions 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. 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 could be dynamically assigned to existing pretrained clusters to make the disambiguation process more efficient for real-time event detection tasks.

References

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

[2] 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.

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

[4] 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*, pp. 3391–3397, 2019.

[5] T. Pellissier Tanon, G. Weikum, and F. Suchanek, "Yago 4: A reasonable knowledge base," *The Semantic Web*, 2020.

[6] 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.

[7] 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*, pp. 229–232, 2011.

[8] 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*, pp. 55–60, 2014.

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

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

[11] 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*, pp. 260–277, Springer, 2017.

[12] 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)*, pp. 29–34, IEEE, 2020.

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

[14] 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*, pp. 1625–1628, 2010.

[15] 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.

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

[17] 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.

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

[19] 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*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

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

[21] 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)*, pp. 997–1002, IEEE, 2014.

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

[23] 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)*, pp. 105–108, IEEE, 2005.

[24] 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, p. P10008, 2008.

[25] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[26] 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, pp. 54–60, 2014.

[27] 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.

[28] B. Locke and J. Martin, "Named entity recognition: Adapting to microblogging," *Senior Thesis, University of Colorado*, 2009.

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