2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Implicit Entity Linking through Ad-hoc Retrieval

Hawre Hosseini*, Tam T. Nguyen†, Ebrahim Bagheri*
*† Laboratory for Systems, Software and Semantics (LS^3), Ryerson University
Email: *{hawre.hosseini,bagheri}@ryerson.ca, †nthanhtam@gmail.com

***Abstract*—The systematic linking of explicitly-observed phrases within a document to entities of a knowledge base has already been explored in a process known as *entity linking*. The objective of this paper, however, is to identify and entity link those entities that are not mentioned but are implied within a document, more specifically within a tweet. This process is referred to as *implicit* entity linking. Unlike prior work that build a representation for each entity based on its related content in the knowledge base, we propose to perform implicit entity linking by determining how a tweet is related to user-generated content posted online and as such indirectly perform entity linking. We formulate this problem as an *ad-hoc document retrieval* process where the input query is the tweet, which needs to be implicitly linked and the document space is the set of user-generated content related to the entities of the knowledge base. We systematically compare our work with the state-of-the-art baseline and show that our method is able to provide statistically significant improvements.**

I. Introduction

Entity linking, which is the task of linking a recognized named entity mention in a given textual piece to a unique entry of a knowledge base, e.g., DBpedia, is now a well studied task for various content types [1]. The main objective of entity linking is to connect between the explicit mentions of entities in a given input text and their corresponding representations in the knowledge base. Given the short and noisy nature of tweets, the main concept being discussed in the tweet may not be easily recognizable. In other words, while a few entities might be recognized in the tweet, the core idea being discussed in the tweet might not be determined and linked to. The idea of *implicit entity linking*, introduced earlier in [2], is to link tweets to relevant entities that are not mentioned in the text but are core to its understanding.

Effective implicit entity linking requires the consideration of subtle clues within the input tweet. Prior work has focused on using the knowledge graph for building a contextual representation of each input tweet based on the collection of *explicit* entity mentions present in the tweet. In our work, while also benefiting from information of the knowledge graph, we primarily rely on informal social content to perform implicit entity linking. More specifically, we formalize the problem of implicit entity linking as a process of connecting the tweet space to the user-generated content space, i.e., user-written reviews about entities such as movie reviews. Therefore, in order to identify the implicit entity of a tweet, we set out to find a set of reviews on social platforms that are highly relevant and closely representative of the content within the tweet. This way, the entity being described by the retrieved reviews would be the implicit entity described in the tweet. In other words, we formalize the problem of implicit entity linking as an *ad-hoc document retrieval* task where the query is the input tweet and the documents to be retrieved and ranked are the set of most relevant reviews to that tweet. To the best of our knowledge, our work is one of the first to employ unstructured social feedback content, i.e., reviews, in the process of entity linking.

IEEE/ACM ASONAM 2018, August 28-31, 2018, Barcelona, Spain
978-1-5386-6051-5/18/$31.00 © 2018 IEEE

II. Proposed Approach

Our approach is comprised of two main steps, namely (1) *candidate selection* and (2) *candidate ranking*.

A. Candidate Selection

The first step in our work is to narrow down the entity search space by selecting a set of candidate entities, which have a higher probability of being related to the tweet of interest. We make use of the DBpedia knowledge graph as the entity search space. Given a Tweet *t*, and the type of the implicit entity that is being sought ϑ, e.g., `wikidata:Film`, the objective of the candidate selection method is to retrieve a set of entities $S = \{s_1, ..., s_n\}$ from DBpedia relevant to *t* and of type ϑ. To this end, we first extract the explicit entities that are present in the input tweet using a standard entity linking tool, e.g., TagME. Once the explicit entities are extracted, we query the DBpedia knowledge graph for all those triples whose subject (or object) match one of the identified explicit entities and the object (or subject) has `rdf:type` ϑ. All such retrieved entities of type ϑ are included in a candidate entity set S.

Due to the short length as well as the informal language of tweets, it is possible that we face the problem of *entity sparsity*. To overcome this issue, we perform *context expansion* for candidate selection where a set of relevant tweets to the tweet of interest are pooled and added as context. In order to perform the search, we look for those tweets that include the surface form of the explicit entities observed in *t* and also have a mention of the `dbp:label` for ϑ, e.g., `wikidata:Film dbp:label` Film. From among the retrieved tweets that match the search criteria, we only retain those that are within a two week time frame from *t* and have at least one explicit entity when ran through an entity linking system.

B. Candidate Ranking

Having found the set of candidate entities S for a given Tweet t, the objective of the disambiguation phase is to rank the entities in S based on their relevance to the tweet of interest. One of the novelties of our work is that we rank entity relevance to the tweet not based on the similarity of the knowledge graph representation of the entity to the tweet but rather through measuring the relevance of the tweet to the social content available for each candidate entity.

Specifically, for each entity s in S, we identify user-generated content related to s that could be used to measure relevance of the tweet to the entity. Given the comparability of the two spaces, the problem of ranking the entities in S based on their relevance to t can be formulated as an ad-hoc retrieval process where by t is an input query and the social content relevant to the entities in S form the document space.

We base the entity ranking process on ad-hoc retrieval methods that are based on Markov Random Fields and show how term dependency, explicit entity and neural embedding features can be embedded in this framework.

1) Markov Random Fields: Markov Random Fields (MRFs) are generally used for modelling joint distributions. In the context of our work, given a tweet t and a review R, the objective is to compute the joint distribution $P(t, R)$. In MRF, a graph G is formed by nodes representing random variables, i.e., tweet terms t_i and a review R, and the edges determine the dependence between them. The joint probability between pairs of random variables t and R is computed as:

$$P_\Lambda(t, R) = \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \phi(c; \Lambda) \tag{1}$$

where R is a review, $C(G)$ is the set of cliques in G, and $\phi(c; \Lambda)$ is a non-negative function. The function is parameterized by Λ and the feature function, denoted by $f(c)$. The normalization factor Z_Λ is ruled out since it is computationally expensive and thus the ranking equation is simplified:

$$P(R|t) \overset{rank}{=} \sum_{c \in C(G)} \boldsymbol{\lambda} f(c) \tag{2}$$

There are many possibilities as to what each potential function can take into account. Different models varying in the feature functions have been proposed for a range of retrieval tasks [3], [4], [5]. We are specifically interested in the *Sequential Dependence Model (SDM)* and *Entity Linking incorporated Retrieval (ELR)* [6] as they systematically consider term dependency and explicit entity mentions in their feature functions, respectively. We further propose a third type of feature function based on neural embeddings, called *Neural Embedding-based Measure of Similarity (NEMS).*

2) Sequential Dependence Model: The Sequential Dependence Model (SDM), which is an effective and efficient MRF-based retrieval model, assumes that tweet terms are sequentially dependent on each other. Following that assumption, tweet terms which are adjacent or in close proximity are connected to each other in the MRF-based graph G. The two-cliques formed between tweet terms and a review give way to three feature functions based on: (1) cliques formed by a tweet unigram (term) and a review node, and (2) cliques involving a node containing two or more ordered terms of a tweet and a review, and (3) cliques involving a node containing two or more unordered terms of a tweet and a review.

3) Explicit Entity Feature Function: As mentioned earlier, explicit entities play an important role in identifying candidate entities from the input tweet in our work. For this reason, we additionally employ a feature function based on explicitly observed entities involving two-cliques formed between the explicit entity mentions of the tweet and the review. The feature function is inspired by the Entity Linking incorporated Retrieval (ELR) framework [6]. This feature function, denoted as $f_E(e, R)$, is formulated as:

$$f_E(e, R) = log\left[(1 - \alpha_R) tf_{\{0,1\}(e,R)} + \alpha_R \frac{tf_{e,R}}{|C_e|}\right] \tag{3}$$

where $tf_{\{0,1\}(e,R)}$ denotes existence of the entity in the review, $tf_{e,R}$ denotes the frequency of the entity in the review, and $|C_e|$ denotes the total occurrence of the entity in the collection. We integrate the confidence score of the entity tagged by the linker, denoted as $s(e)$ into the probability equation for this clique type to obtain the feature function in the following form:

$$P(R|t) = \sum_{c \in C(G)} \lambda_E s(c) f_E(c, R) \tag{4}$$

4) Neural Embedding-based Feature Function: One of the limitations of the ELR feature function is that if an entity in t, or other reachable entities from t, do not occur in R, the value of the feature function would be zero. However, in practice, an effective feature function needs to be able to identify cases where while an entity c_1 in t does not appear in R, some closely related entity c_2 is mentioned in t. To overcome this limitation, we propose a neural embedding-based feature function to capture the relationship between entities even when they are not observed in the tweet or reviews. The objective of the *neural embedding learning* model is to learn vector representations that are useful for predicting entity relevance. Different from the vanilla word2vec models where word neighborhood is used for learning word similarity, we formulate our neural embeddings to learn the similarity at phrase and entity levels. Formally, a sentence with a list of words $w_1, w_2, ..., w_n$ and their corresponding part-of-speech tags $t_1, t_2, ..., t_n$, can be related to a list of entities $e_1, e_2, ..., e_m$ where $e_i = [(w_k, t_k)\ , ..., (w_{k+l-1}, t_{k+l-1})]$ is an entity with a surface form length *l*. Our objective is to maximize the log probability as follows:

$$\frac{1}{m} \sum_{i}^{m} \sum_{(j \in S) \cap (j \neq 0)} log\ p(e_{i+j}|e_j) \tag{5}$$

where $log\ p(e_{i+j}|e_j)$ is the skip-gram softmax function and S is the context window. Normally, each entity is assigned a unique identifier and encoded in a sparse vector space. The special characteristic of our neural embedding model is that it jointly considers word neighborhood, part-of-speech tag information and entity relations in the neural embedding.

Now given a trained neural embedding model, we expand the tweet t by searching for similar entities in the model and selecting top-*k* entities to retrieve relevant reviews. Formally,

TABLE I
THE RECALL OF THE CANDIDATE SELECTION PHASE.

	Domain	Without Expansion	With Expansion
Our Approach	Movie	0.75	0.81
	Book	0.72	0.8
Baseline	Movie	0.77	0.9
	Book	0.76	0.94

we define a novel embedding-based feature function as follows. Let $v_1, v_2, ..., v_m$ be the vector representations of entities $e_1, e_2, ..., e_m$, we have the kernel matrix of these entities:

$$M(v_i, v_j) = v_i \cdot v_j \tag{6}$$

For each entity e_i, we find a list of most relevant entities:

$$E(v_i, k) = \{e_i\} \cup \{e_j | argmax_{j \neq i} (M(v_i, v_j), k)\} \tag{7}$$

Based on the list of most relevant entities, the Neural Embedding-based Measure of Similarity (NEMS) feature function is defined as:

$$f_N(e, R) = \sum_{c \in E(e,k)} f_E(c, R) \tag{8}$$

where f_N is the feature function based on the two-cliques formed between most relevant entities to the explicit entities of a tweet and the review R and F_E is from ELR.

5) Interpolation for Learning Feature Weights: In MRF, the final ranking function is defined as a linear interpolation of all feature functions: $P(R|t) \overset{rank}{=} \sum_{\eta \in \{SDM,E,N\}} \sum_{c \in C(G)} \lambda_\eta f_\eta(c, R)$ where λ_η is a weight vector $\boldsymbol{\lambda}$. Let $\boldsymbol{f}$ be a feature vector of all c, we have a vector form of the final ranking function:

$$P(R|t) \overset{rank}{=} \boldsymbol{\lambda} \cdot \boldsymbol{f} \tag{9}$$

In practice, the relationship of these feature functions may not be linear. In order to make the interpolation more realistic and improve the performance of the ranking function, we extend it by mapping these feature functions into higher dimensional space using *polynomial kernel*. Therefore, our newly proposed ranking function is defined as follows:

$$P(R|t) \overset{rank}{=} \boldsymbol{\lambda} \cdot \gamma(\boldsymbol{f}) \tag{10}$$

where $\gamma(\boldsymbol{f})$ is polynomial kernel. To learn the weight vector in polynomial space, we apply the stochastic gradient descent algorithm. In our experiments, we used log loss function, L2 norm, and a polynomial kernel. The learning rate was set to $1/(0.001(t + t_0))$ where t is the number of data points we have seen and t_0 is a smoothing parameter.

III. EXPERIMENTS

In order to evaluate our work, we adopt the evaluation methodology, gold standard dataset and evaluation metrics proposed by Perera et al. [2] and use the method proposed in their paper as the baseline. In the baseline, the authors have focused on two domains, namely *books* and *movies* for which 207 and 190 tweets with an average length of 18 words were identified and manually labeled, respectively. The gold standard dataset is available at https://goo.gl/jrwpeo. As suggested in the baseline, the evaluations include both the performance of candidate selection and candidate ranking steps *individually* and *in tandem*. Furthermore, given the gold standard collection of tweets is related to 2014 tweets and entities, we crawled all the reviews related to books and movies published in 2014 from GoodReads and RottenTomatoes, which consisted of $3,181$ books and 956 movies, respectively. These reviews were then used to form the review space used in the implicit entity linking process. For the set of movies, we collected all the entities on DBpedia of type Film that were released in 2014. We then used the corresponding Wikipedia URL for that entity to scrape its RottenTomatoes URL, which was then used to collect all the relevant reviews. We performed a similar process where the set of all books published in 2014 were collected from DBpedia, whose ISBNs were used to reach them on GoodReads and hence the related reviews were collected.

Furthermore, given our proposed feature function (NEMS) and the feature function from ELR depend on the explicit entities available in the input tweet as well as the reviews, we performed explicit entity linking on the input tweets and reviews using two entity tagging systems, namely TagMe [7] and AIDA [8] to link tweets/review to DBpedia concepts. The reason we chose two taggers was to show the impact of the entity tagger performance on the overall performance of our work. We have posted our code, annotations of tweets and reviews based on both TagMe and AIDA on the following Github repository: https://goo.gl/Cp1YkQ for reproducibility purposes. In the experiments, we first report the results based on TagMe and compare the results with the baseline and then compare the performance of our method using TagMe against AIDA to study the impact of the tagger performance. It is important to mention that when training NEMS, the hyperparameters were context window size S of 5, negative sample of 5 and an embedding dimension size of 128.

Table I shows the recall of the candidate selection method with and without context expansion. As seen in the table, there are three observations that can be made in terms of candidate selection. First, both our approach and the baseline perform quite competitively in terms of candidate selection when context expansion is not applied. Second, both approaches show improved candidate selection recall when context expansion is applied. Third, our approach shows a weaker recall rate in the selected candidates compared to the baseline after context expansion. The lower recall in candidate selection is due to enforcing stricter policies for selecting tweets that are pooled such as ensuring that tweets have at least one linked entity. We will show that this weaker performance in recall will enhance retrieval accuracy and precision.

We further analyze the performance of the candidate ranking process. Given the MRF retrieval model adopted in our work, the quality of the produced ranking is dependent on the feature functions used in our work. We compare the performance of the three main variations of the retrieval models with the baseline, i.e., SDM, the interpolation of SDM and ELR, as well as the interpolation of SDM, ELR and NEMS. We report the results for both cases when uniform values are used for the in-

TABLE II
THE PRECISION@1 OF THE RANKING METHODS. ▲ SHOWS STATISTICAL SIGNIFICANCE OVER ALL OTHER METHODS AT 0.05 WITH A PAIRED T-TEST.

Domain	Movies		Books	
	Uniform Params	Trained Params	Uniform Params	Trained Params
SDM	0.25	0.61	0.14	0.46
SDM+ELR	0.38	0.71	0.33	0.53
SDM+ELR+NEMS	0.57▲	0.77▲	0.61▲	0.75▲
Baseline	n/a	0.61	n/a	0.61

TABLE III
THE ACCURACY OF THE IMPLICIT ENTITY LINKING METHOD COMPARED TO THE BASELINE.

	Candidate Selection		Candidate Ranking		Overall Accuracy	
	Baseline	Our Approach	Baseline	Our Approach	Baseline	Our Approach
Movie	0.90	0.81	0.61	0.77	0.55	**0.62**
Book	0.95	0.80	0.61	0.75	0.57	**0.60**

TABLE IV
THE RECALL OF THE CANDIDATE SELECTION PHASE (TAGME VS AIDA).

	Domain	Without Expansion	With Expansion
TagMe	Movie	0.75	0.81
	Book	0.72	0.8
AIDA	Movie	0.62	0.69
	Book	0.58	0.65

TABLE V
THE PRECISION@1 OF SDM+ELR+NEMS BASED ON TAGME VS AIDA.

Domain	Movies	Books
	Trained Params	Trained Params
TagMe (SDM+ELR+NEMS)	0.77▲	0.75▲
AIDA (SDM+ELR+NEMS)	0.66	0.59

terpolation coefficients and when interpolation coefficients are trained in Table II. We find that the SDM+ELR+NEMS model outperforms the baseline as well as SDM and SDM+ELR on precision@1 and the improvement is statistically significant (paired t-test at p-value < 0.05). The improvement of this model over SDM+ELR shows that the neural embedding feature introduced in this paper significantly impacts the ranking performance. This is in line with earlier findings that neural embedding features can enhance retrieval performance in the context of learning to rank methods [9] and ad-hoc retrieval [10], [11]. Table III reports the overall accuracy of the implicit entity linking process. It should be noted that the recall of the candidate selection step is not comparable to P@1 of the ranking method because these are the performances of two separate steps. The overall performance of our method and the baseline, when considering both steps, would be the product of the performance of each step, reported in Table III. As seen in the table, our approach shows statistically significant improvement compared to the baseline on both domains.

As mentioned earlier, we also evaluated the impact of the tagger performance on the overall performance of our method. To this end, we use another state-of-the-art tagger, AIDA [8], to compare with the performance of TagMe. In our experiments, we found that AIDA results in worse performance compared to TagMe. Summarily, based on Table IV, AIDA vs TagMe reported 0.62 vs 0.75 and 0.69 vs 0.81 (without and with expansion for the movie dataset) and 0.58 vs 0.72 and 0.65 vs 0.80 (without and with expansion for the book domain). Also based on Table V, AIDA vs TagMe reported a P@1 of 0.66 vs 0.77 and 0.59 vs 0.75 for movie and book datasets. This is in line with earlier studies that show TagMe is one of the better performing tagging systems for short text such as tweets [12].

IV. CONCLUDING REMARKS

In this paper, we proposed an approach for performing implicit entity linking on tweets. Our work is novel in that (1) we do not directly link a tweet to the knowledge graph entity of interest and instead indirectly link the tweet based on its similarity to online user-generated content; (2) we formulate the problem of implicit entity linking as an ad-hoc document retrieval task where the ranking of user-generated content for a tweet determine the relevant implicit entity; and (3) we propose a new neural embedding-based feature function and incorporate it into the MRF-based retrieval framework. We have shown, based on a comparable gold standard, that our method outperforms the baseline in precision and accuracy.

REFERENCES

[1] W. Shen, J. Wang, and J. Han, “Entity linking with a knowledge base: Issues, techniques, and solutions,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 443–460, 2015.

[2] S. Perera, P. N. Mendes, A. Alex, A. P. Sheth, and K. Thirunarayan, “Implicit entity linking in tweets,” in *European Semantic Web Conference 2016*, 2016, pp. 118–132. [Online]. Available: https://doi.org/10.1007/978-3-319-34129-3_8

[3] D. Metzler and W. B. Croft, “Latent concept expansion using markov random fields,” in *SIGIR 2007*, 2007, pp. 311–318.

[4] ——, “A markov random field model for term dependencies,” in *SIGIR 2005*, 2005, pp. 472–479.

[5] F. Ensan and E. Bagheri, “Document retrieval model through semantic linking,” in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, 2017, pp. 181–190. [Online]. Available: http://dl.acm.org/citation.cfm?id=3018692

[6] F. Hasibi, K. Balog, and S. E. Bratsberg, “Exploiting entity linking in queries for entity retrieval,” in *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ser. ICTIR ’16. New York, NY, USA: ACM, 2016, pp. 209–218. [Online]. Available: http://doi.acm.org/10.1145/2970398.2970406

[7] P. Ferragina and U. Scaiella, “Fast and accurate annotation of short texts with wikipedia pages,” *IEEE Software*, vol. 29, no. 1, pp. 70–75, 2012. [Online]. Available: http://dx.doi.org/10.1109/MS.2011.122

[8] Y. Ibrahim, M. Amir Yosef, and G. Weikum, “Aida-social: Entity linking on the social stream,” in *ESAIR ’14*, 2014, pp. 17–19.

[9] F. Ensan, E. Bagheri, A. Zouaq, and A. Kouznetsov, “An empirical study of embedding features in learning to rank,” in *CIKM*.

[10] H. Zamani and W. B. Croft, “Relevance-based word embedding,” in *SIGIR 2017*, 2017, pp. 505–514.

[11] E. Bagheri, F. Ensan, and F. Al-Obeidat, “Neural word and entity embeddings for ad hoc retrieval,” *Inf. Process. Manage.*, vol. 54, no. 4, pp. 657–673, 2018.

[12] M. Cornolti, P. Ferragina, and M. Ciaramita, “A framework for benchmarking entity-annotation systems,” in *Proceedings of the 22Nd International Conference on World Wide Web*, 2013, pp. 249–260.