Scalable graph-based individual named entity identification

Sammy Khalife
Ecole Polytechnique, LIX, Dascim
Palaiseau F-91120
khalife@lix.polytechnique.fr

Michalis Vazragiannis
Ecole Polytechnique, LIX, Dascim
Palaiseau F-91120
mvazirg@lix.polytechnique.fr

ABSTRACT

Named entity discovery (NED) is an important information retrieval problem that can be decomposed into two sub-problems. The first sub-problem, named entity recognition (NER), aims to tag pre-defined sets of words in a vocabulary (called "named entities": names, places, locations, ...) when they appear in natural language. The second sub-problem, named entity linking/identification (NEL), considers these entity mentions as queries to be identified in a pre-existing database. In this paper, we consider the NEL problem, and assume a set of queries (or mentions) that have to be identified within a knowledge base. This knowledge base is represented by a list of the possible spelling of named entities, moreover their text recognition (NER) - is not trivial since we do not have an exhaustive set of words that represent it can also be referred as a named entity (or mention), and require lots of parameter tuning along with large volume of annotated data.

First of all, we propose a filtering algorithm designed with information retrieval and text mining techniques, aiming to maximize precision at K (typically for $5 \leq K \leq 20$). Then, we introduce two graph-based methods for named entity identification to maximize precision at 1 by re-ranking the remaining top entity candidates. The first identification method is using parametrized graph mining, and the second similarity with graph kernels. Our approach is well-motivated by the limitations brought by recent deep learning approaches that lack interpretability, and require lots of parameter tuning along with large volume of annotated data.

Finally, we conclude on the advantages of our graph-based approach compared to recent deep learning methods.

KEYWORDS

Information retrieval, named entities, graph mining, graph kernels, supervised learning

1 INTRODUCTION

1.1 Basic Concepts

The purpose of Named entity discovery (NED) in machine learning and natural language processing is two-fold. First, it aims to extract pre-defined sets of words from text documents. These words are representations of named entities (such as names, places, locations, ...). Then, these entity mentions paired with their context are seen as queries to be identified in a pre-existing database [16]. Firstly, it is important to stress that the subtask of NED - Named entity recognition (NER) - is not trivial since we do not have an exhaustive list of the possible spelling of named entities, moreover their text representation can change (for example, "J. Kennedy" vs. "John Kennedy").

In this paper we focus on the second task, Named entity linking (NEL). Let us define it properly.

Named entity (and Mention/Query): An entity is a real-world object. It usually has a physical existence, but can be abstract. It is denoted with a proper name. In the expression "Named Entity", the word "Named" aims to restrict the possible set of entities to only those for which one or many rigid designators stands for the referent [22]. When a named entity appears in a document, the words that represent it can also be referred as a mention. Finally, a query refers to the mention, the context where it appears, and associated type. (We give more explanation on the notion of type in the Knowledge base definition.)

Example: "John Kennedy served at the height of the Cold War". In this sentence, John Kennedy is a named entity (or mention), and the associated query is the name "John Kennedy", the sentence, and the named entity type (e.g Person).

Knowledge base/graph: A Knowledge base is a database providing supplementary descriptive and semantic information about entities. The semantic information is contained in a knowledge graph, where a node represents an entity, and an edge represents a semantic relation. In the general case, the knowledge graph can be of any kind (directed, weighted, ...). See figure 1 for an example. We discuss knowledge graph types in details in part 3 and Evaluation.

Figure 1: Representation of a unweighted directed semantic graph (Wikipedia/NIST TAC-KBP Challenge 2010). An edge between two entities $E_1$ and $E_2$ represents a link from $E_1$ to $E_2$.

Usually, entities have a type feature [22], such as : PER, ORG and GPE (respectively person, organization and localization). For instance, the entity "United States" is a GPE, John Kennedy is a PER, etc... These types play a central role in NER and NEL. Indeed, these features decrease the number of candidate entities for identification.
However, it is possible to consider a fine-grained classification, with hundreds of entity types, similarly to DBpedia ontology\(^1\).

**Named entity linking (NEL):** Given mentions of entities in digital data (named entities, or mentions) the purpose of named entity linking is to identify the corresponding unique (one entity per mention) ground truth entities (also referred as gold entities) in a database (knowledge base).

1.2 Contributions

In this work, we provide an overview of the NEL problem, and investigate two graph-based methods. We highlight their advantages and limitations over recent deep learning approaches.

In the following, the first step, referred as entity filtering, aims at reducing entity candidates to top $K$ entities for one query. The second step, referred as entity identification, aims at identifying the true entity among the remaining $K$ candidates, for which we propose two graph-based routines (selective graph mining and graph comparison with kernels). We present the construction of our algorithms in details, along with their computational complexity. We also include an evaluation of experimental performance on several datasets, with an analysis of the impact of parameter $K \in \{5, ..., 20\}$, and a detailed comparison with existing approaches. We do not include in this work the problem of Fine-grained named entity recognition [21] (word tagging) nor NIL-detection (detection of entities out of the knowledge base).

2 RELATED WORK

In general, linking between named entities and a knowledge graph can be done either individually or collectively. In the first case individual mentions are considered separately, as independent queries. In the collective way, we consider dependence between queries in a same document, where the true entities should have some proximity or coherence. Therefore, a collective linking framework implies some dependence between these entity variables. For the sake of completeness, we report here both individual and collective approaches. In the following subsections, we present 3 categories of state-of-the-art algorithms for named entity linking.

**Notations :** $E = \{1, ..., E\} \subset \mathbb{N}$: indexes of entities and $M = \{1, ..., M\} \subset \mathbb{N}$: indexes of mentions, $\hat{e}_i$: system’s output entity index for mention index $m_i$.

2.1 Graphs for NED

Individually collective linking : Given a real value scoring function defined on the product space of mentions and entity states (for example, combinations of Jaccard index over N-grams), let $W_{i,j}$ the corresponding score between the mention $i$ and the entity $j$. For individual disambiguation, one wants to perform independent mention-entity attribution. Then the graph structure is irrelevant and the formulation is straightforward:

$$\hat{e}_i = \text{arg max}_{j \in E} W_{i,j} \quad (1)$$

In the collective linking formulation, the optimization formulation is different: the underlying gold entities should respect some arbitrary semantic coherence. The coherence information is represented within a coherence function $\psi : E^M \rightarrow \mathbb{R}$ between the entity candidates. Usually $\psi$ is defined using knowledge graph structure. For example $\psi$ can be defined as the shortest-path function on the knowledge graph. With these notations, the set of selected entities are formally defined as:

$$\hat{e}_1, ..., \hat{e}_M = \text{arg max}_{j_1, ..., j_M \in E^M} \left( \sum_{i=1}^{M} W_{i,j_i} + \psi(j_1, ..., j_M) \right) \quad (2)$$

Equation 2 can be formulated as a boolean integer program, but the nature of $\psi$ being arbitrarily complex (e.g. shortest-path function) does not allow to solve the general case, especially when $M \rightarrow +\infty$. Therefore, other formulations are preferred: a rule-based individual linking has been proposed [14], and [17] proposed a collective formulation for entity linking decisions, in which evidence can be reinforced into high-probability decisions.

Other formulations using community detection and Pagerank have been proposed, for which we give detailed explanations in the next paragraphs.

**Bipartite graph & community detection :** Similarly, we can model NEL as a bipartite graph optimization problem. One of the nodes set is built using the knowledge graph: the graph can be directed or undirected, and weighted (using similarity functions for instance). In the entity nodes set, the information is structured, clear, canonic and considered true. The second nodes set are the queries and contain potentially ambiguous information (cf figure 2).

![Figure 2: Directed and undirected query/entity bipartite weighted graph. For each graph, the set on the right is built using an extract of the knowledge graph (same extract as figure 1): Nodes $e_1$, $e_2$, $e_3$ are entities the knowledge base. Nodes $a$ and $b$ are entity mentions extracted from text documents forming queries.](image)

In this context, graph-based approaches have been developed. [18], and [2] proposed to link efficiently mentions to their corresponding entities using the weighted undirected bipartite graph built among mentions-entities text similarities, by extracting a dense subgraph in which every mention node is connected to exactly one entity, yielding the most likely disambiguation.

In general, this combinatorial optimization problem is NP-hard with respect to the number of nodes, since they generalize Steiner-tree problem [18]. However, heuristics have been brought forward, such as [18] and [2] proposing a discarding algorithm using taboo search and local similarities with polynomial complexity.
PageRank: Another graph-based proposal is to use an adaptation of PageRank algorithm to provide entities a popularity score. For example, [31] built a weighted graph $G$ of all mentions and entities based on local and global similarities, and capitalize on the Hyperlink-Induced Topic Search (HITS) algorithm to produce node authority scores. on the graph. Then, within similar entities to mentions, only entities with high authority will be retained.

These graph-based algorithms proved to be fast compared to the other approaches since they do not require any training, and perform reasonably well in terms of precision. However, most of them are greedy and need pre-filtering to discard some entity candidates.

2.2 Probabilistic graphical models
An interesting idea is to consider mentions as random variables and their golden/true entities as hidden states. Unlike character recognition where $|E| = |E_i| = 26$ for latin alphabet, the number of possible states per entity - usually $\geq 10^6$ - and Viterbi algorithm quadratic complexity $O(N |S|^2)$, where $S$ is the number of states and $N$ the number of observations) makes the problem computationally untractable. To overcome this technical issue, a first step proposed by [1] is to establish a reduced set of candidates per mention: $m_i \in E_i$ using mention context. Using annotation, an HMM is trained on the reduced set of candidates. Inference is made using message passing (Viterbi algorithm) to find the most probable named entity sequence. Another approach using probabilistic graphical model has been provided by [9], with a factor graph that uses popularity-based prior, Bethe Approximation to decrease inference computational cost, and message passing to compute marginal probabilities. The computational complexity is $O(N^2 r^2)$ where $r$ the number of average entity candidates per mention and $N$ the number of observations.

Finally, another probabilistic graphical model has been proposed, similarly to latent Dirichlet allocation (LDA) [5], where an iterative procedure $P$ is used above the LDA-scheme to enrich the knowledge base. Its complexity is proportional to the product between LDA complexity and the number of iterations of procedure $P$ [20].

A disambiguation tool using pre-trained embeddings, then averaging and ranking has been proposed [33] with a $O(m e^2)$ complexity, where $m$ is the number of mentions and $e$ the number of entities.

Recent advances in neural networks conception suggested to use word embeddings and convolutional neural networks to solve the named entity linking problem. [30] proposed to maximize a corrupted cosine similarity between a mention, its annotated gold entity and a false entity. The network is trained with polynomial complexity, and reached state-of-art performance in precision (until 2017 and [34]) on NIST TAC-KBP datasets in 2009 and 2010.

Long-short-term memory networks (LSTMs) recently provided remarkable results for natural language modeling in general. Recent neural network architecture have been proposed [26, 29], the latest using a recent method using fine-grained ontology type system and reaching promising results on several datasets.

2.4 Comments
Each of the cited approaches uses a filtering metric to discard non relevant entities. To the best of our knowledge, this is the case for every state-of-the-art routine. Therefore, the final precision score will be upper-bounded by the recall of the filtering, shown on figure 3. it is widely accepted that neural networks require very large datasets. Moreover, it is data quality of automatic generated mentions from Wikipedia is debatable.

![Figure 3: Visualization of a NEL workflow. Entity linking is performed in 2 steps.](image)

3 METHODOLOGY
Due to the aforementioned comments, we chose to investigate graph-based methods along with supervised learning algorithms requiring a reasonable amount of data. This section addresses entity filtering and identification. In the first subsection, we explain the experimental factors causing a drop in performance for entity filtering. Based on this analysis, we propose a new entity filtering method using information retrieval techniques. In the second subsection, we present two new graph-based methods for entity identification.

All our method is conceived for individual linking: queries are considered separately.

3.1 Entity filtering
Assume the following document sentence: "J. Kennedy has met soviet Premier Nikita Khrushchev and Kennedy in 1961, in Vienna".
A named entity recognizer will tag "J. Kennedy" as a mention. We would like to discard wrong entity candidates using named entity filtering. To do so, we consider only two sources of information in the query: the mention name ("J. Kennedy"), and the information contained in the rest of the document, i.e. the other words. The filtering precision is an upper bound of the overall precision of the NEL workflow displayed in figure 3.

3.1.1 Example of entity filtering.
Given a filtering algorithm $F$, let $E_{GM}$ be the generic golden entity random variable of mention $M$. Let $p_F$ be the following mention/entity prior:

$$p_F = \mathbb{P}(E_{GM} \in F(M))$$  \hspace{1cm} (3)

A performing filtering method should maximize this prior. In information retrieval, and especially for NEL, one of the most popular metric is built using combinations of $N$-grams. [4, 8, 17, 23]. Considering these sets of $N$-grams, the Jaccard index is a real value defined by the ratio between the size of the intersection of these sets divided by the size of the union of the sets. Let us denote $F_{NK}$ the filtering algorithm (here $K$ top results) using Jaccard index score using $n$-grams. We suppose the type of the query is known in advance (PER, ORG, GPE), and consider entity candidates accordingly.

We compute the prior parameter estimation (here $n$ is the number of samples):

$$\hat{p}_{FNK} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{E_{GM_i} \in F_{NK}(M_i)\}}$$  \hspace{1cm} (4)

over 2010 NIST TAC-KBP dataset (we refer to experiments section for a description of the datasets). Using mean of Jaccard index over 2-grams, 3-grams and 4-grams, and $K = 50$, it turns out that $\hat{p}_{FNK} = 0.83$. This implies that one cannot hope to reach a better empirical precision at 1 that 83% if we chose this filtering method. A low value of $p_{FNK}$ causes a greedy drop of precision. Therefore, it is important to maximize $R@K$ in the first place.

3.1.2 Experimental explanation.
In this subsection we emphasize factors causing drop in precision with the previous filtering example.

Acronyms: Acronyms played an important role in entity filtering. For example, in NIST TAC-KBP 2009 dataset, "gsu" represents Georgia state university, "ccp" is used for Communist party of China, "abc" for Australian broadcasting corporation, "cdu" for Christian Democratic Union (Germany).

Nicknames, other names, ancient names: Geographical entities such as cities and countries have some historical background, and can be referenced by a query that is not similar to the ground truth entity name. For example, "Beehive state" refers to Utah state in the United State of America, "Flavia Neapolis" or "Little Damascus" or "Shechem" refer to Naplouse in Cisjordanie, "Garden city" to Port Harcourt in Nigeria. Other entity types such as people are concerned: "Iron Lady" refers to Margaret Thatcher, ex-Prime Minister of United Kingdom.

String comparison is not enough: String comparison sometimes fails due to overlapping of mentions and entities names. This becomes a problem when the number of entities in the knowledge base is high. For example, Wikipedia knowledge base contains millions of entities (cf. experiments section). In this case, Jaccard distance over combinations N-grams cannot capture resolve ambiguity. For example, "State of Utah" has as 1st ranked entity: "Nevada State Route 531", whereas its gold entity is "Utah".

3.1.3 Our filtering method:
In order to improve previous filtering algorithms, we propose a routine based on four main components: preprocessing, acronym detection and expansion, name scoring and context scoring. In practice, pre-processing is applied first on the data. Then, the three remaining steps (gathered in algorithm 1) are performed.

a - Preprocessing:
For trivial queries having a mention name equal to an existing entity name, we implemented a naive match pre-processing. This is performed by saving a dictionary which keys are the names of knowledge base entities, and values the actual entity ID.

b - Acronym detection/expansion
(refered as ACRD in algorithm 1).
Acronym detection and expansion is a common topic in bioinformatics. We refer to [7] as a survey of acronym detection methods.

- Detection: Following [35], we explored a supervised learning approach, but chose simple rule-based decision based on the string length and cumulated length between each capital letter [15].
- Expansion: The score is the length of longest common substring [3] between acronym string and capital letters of the entity target

c - Name scoring (Refered as JN in algorithm 1)
When the named entity (mention) is not tagged as an acronym, comparison with entity titles is performed:

- Compute $N$-grams for $N \in \{2, 3, 4\}$
- Average Jaccard Index of mention name and entity title. [definition of Jaccard index in 3.1.1]

d - Context scoring: (refered as $tfdifScore$ in algorithm 1)
This part aims to maximize similarity of sentences. We experimented several techniques:

- Keyword detection (by frequency) and intersection
- Document distances using word embeddings, especially Word Mover's Distance [19]
- Cosine similarity of TF-IDF vectors. The matrix is computed on all the knowledge base. It is refered as $tfdifScore$, and The scoring algorithm is summarized in Algorithm 1.

We decided to keep cosine similarity of TF-IDF vectors for three reasons. First, for experimental reason: we did not see any significative difference in performance using word embeddings. Second, this choice seemed more consistent with the motivation of this paper to challenge recent approaches using such embeddings. Finally, we wanted to propose a straightforward candidate generation method that can be implemented in a real system, without any annotated training data.

The scoring algorithm for entity filtering is summarized in Algorithm 1. $T$ is the named entity considered, given as input in the data. Its linear computational complexity with respect to the number of queries and entities is immediate (reminded in part 3 and 4). (All the code is included in our code repository (available on demand)).
Algorithm 1 Entity filtering (generation of entity candidates)

Require: Parameter $K$, Query ($Q=\{M, C, \hat{T}\}$), Entities ($E_T, T$)$\forall E_T, E_S$.

1: $ds=\emptyset$
2: $y_{acr} \leftarrow ACRD(M_i)$
3: for $j=1 \rightarrow E$ do
4: \hspace{1em} if $T_j=\hat{T}$ then
5: \hspace{2em} if $y_{acr}=1$ then
6: \hspace{3em} $s_n = acronymScore(M_i, B_j)$
7: \hspace{2em} else
8: \hspace{3em} $s_n = N(M_i, B_j)$
9: \hspace{1em} end if
10: \hspace{1em} $s_t = \frac{1}{2}(s_n + tfidfScore(C_i, B_j))$
11: \hspace{1em} Sorted insertion by value of $\{j : s_t\}$ in $ds$
12: end if
13: end for
14: return $ds[K]$ ($K$ top entities)

3.2 Graph-based ranking algorithms

In this section, we propose two different graph-based procedures for named entity identification. We capitalize on our filtering method (cf previous subsection) to maximize precision at $K$ (we denote it $R@K$, typically with $5 \leq K \leq 20$), in order to get a limited amount of entity candidates. These graph-based methods use enriched features extracted from the knowledge graph, and re-rank these top entity candidates in order to return the ground truth as first ranked entity.

3.2.1 Knowledge graphs structures.

Our methods do not take in account weights for two reasons. First, by definition : edges indicate a semantic relation, and it is difficult to estimate the intensity of such relation in practice. One could think of this intensity as a quantity proportional to the number of occurrences of two entities together, but we do not have access to such information in the knowledge base, where links can appear only once but still be significant (Example : Country to its City capital). A natural way to assign an intensity to each relation would use edge classification, but we did not include it in this paper.

Second, our methods precisely allow graph search independently of edges weights. Similarly, our algorithms do not penalize local search due to global properties of the graph, on the contrary of methods such as PageRank that assign each node a popularity.

Through the experiments of the paper, we assumed unweighted and undirected knowledge graph even though our methods can also be applied to directed graphs.

3.2.2 First method : parametrized graph mining.

A natural idea to take into semantic information is to use graph mining on the knowledge graph. To do so, we propose for one entity candidate to aggregate context scores of relatively "close" nodes in the knowledge graph. This context scores are computed similarly as for the filtering method (TF-IDF cosine similarity).

Graph & node neighborhood : There are several ways to define a node neighborhood in a graph. We implemented two situations : first, direct neighbors, and then using breadth-first-search (BFS) until a distance threshold. We did not see any improvement with the second method, we present here the method with direct neighbors.

Graph exploration creates a sparsity-noise trade-off (sparsity in case of a low numbers of new entities, and noise from irrelevant entities).

To control this trade-off and select a "convenient" set of entity neighbors, we parametrize this neighborhood selection using a type-mapping function. More formally, let $T$ be the set of indexes of ontology types. A type-mapping function is a (symmetric) boolean function defined on couples of entity types :

$$\varphi \in \Phi : T \times T \rightarrow \{0, 1\}$$

$$\{t_1, t_2\} \mapsto \varphi(t_1, t_2)$$

Therefore, the property $\varphi(t_1, t_2) = \varphi(t_2, t_1)$ should be always true. By definition, $\varphi(t_1, t_2) = 1$ if and only if type 1 and type 2 are "jointly" interacting, i.e given one type or the other, the second has to be considered through graph mining.

Example : Cities : In this example indexes from 0 to 4 represent respectively entity types "City", "State", "Museum", "Country", and "FootballPlayer. Let us choose a mapping function such that:

$$\varphi(0, t_1) = \begin{cases} 0 & \text{if } t_1 = 4 \\ 1 & \text{otherwise} \end{cases}$$

i.e we do not consider type interaction between City and FootballPlayer in the knowledge graph. We suppose that information based on country, state museum helps to disambiguate cities [cf figure 4].

Figure 4: Two homonyms : Cambridge cities.
Left : England city. Right : Massachussets, United States.
Following equation 5, a neighbor entity $e$ is in blue if $\varphi(T\text{City}, T_e) = 1$ and yellow otherwise (Wilf Mannion is a former Cambridge football Player in England).

Features extraction : Let $q$ and $e$ respectively a query and one entity. For the sake of simplicity, we denote by $X^\Phi e$ the generic features vector associated with the couple $(q, e)$. $(n_i)_{1 \leq i \leq deg(e)}$ represents the neighbors of $e$, $t_q$ and $t_n$, respective entity types, and a scoring function $s$ (for example, tfidfScore in algorithm 1). With same notations, we define the corresponding features as follows :

$$\forall i \in \{1, \ldots, |T|\}, \quad (X^\Phi e)_i = \begin{cases} s(q, e) & \text{if } i = 1 \\ s(q, t_q)\varphi(t_q, t_n) & \text{otherwise} \end{cases}$$

For a given entity candidate $e$, if $deg(e) < |T|$, then we complete scores by 0 or use graph exploration to complete the score vector. Here, function $\varphi$ plays the role of a hyperparameter.

A sample class is defined as :

$$y_{q, e} = \begin{cases} 1 & \text{if } e \text{ is the gold entity} \\ 0 & \text{otherwise} \end{cases}$$
Supervised NEL training: With this formulation, we can train NEL regressors or classifiers in a supervised learning framework. At inference, the couple \((q, \hat{e})\) maximizing the prediction score yields predicted entity \(\hat{e}\). If same scores are returned for different couples, we return the first candidate. (This situation didn’t occur in practice). For our experiments, we used simple classifiers: regression trees, random forests, and logistic regression (details in experiments section).

Hyper-parameter tuning: Using a boolean formulation to represent hyper-parameter function \(\phi\), its selection can be interpreted as a boolean combinatorial optimization problem. The empirical optimal mapping depends a-priori on the knowledge graph structure and the queries. The evaluation of the training cost function is not “immediate” since one has to extract new-features for each new mapping function.

This challenge combined with a tough combinatorial problem invited us to consider the following “routine”:

- Train NEL systems with trivial \(\phi\) (constant equal to 1)
- Use meta-heuristics methods for hyperparameter optimization, mainly genetic programming [12].

Graph mining and inference procedure: Based on the previous statements and Figure 4, we sum up graph mining routine in algorithm 2. The final inference routine is presented in algorithm 4.

Algorithm 2 Selective graph mining (SGM) method

Require: Knowledge Graph \(G\), Query \(Q\). Entity top candidates with initial filtering score \((E_t, S_t^i)_{1\leq i \leq K}\). Projection types \((T)_{1\leq i \leq |T|}\). Disambiguation map \(\phi\).

1: for \(i = 1\) to \(K\) do
2: \hspace{0.5cm} Get neighbor nodes in the graph \((N_j(E_t))_{1\leq i \leq |T|}\) in \(G\) according to \(\phi\) and type mapping \(\phi\) according to equation 6
3: \hspace{0.5cm} Build scores vector \((S^i_1)\) \(\cup (S^i_{N_j(E_t)})\) for each type according to equation 6
4: end for
5: return Score vectors \((S^i)_{1\leq i \leq K}\)

3.3 Second method: Graph similarity with kernels

The main idea of this method is to use another graph representation of entities, and use another similarity measure than scoring function \(s\) used in equation 6 (e.g. tfidf).

Graph of Words: Graph of words (GoW) is a representation built over a sequence of objects in order to capture sequential relationships. It has proven its efficiency for several information retrieval problems [27]. Given a window size, nodes are added to the graph by their string representation. Edges are added between nodes in the same sliding window. Figure 5 shows an example with text (window size = 4).

For example, it has been shown [27] that k-core on the graph-of-words representation yield excellent keywords extraction. Using this algorithm, is possible to compute a graph of word representation of a query and the definition of an entity.

3.3.1 Graph similarity: To compare these graph representations, several methods are available. We remind briefly two of them and compare query graph and select the most adapted to the NEL problem.

Sub-graph isomorphism: Given two graphs \(G\) and \(H\) input, and one must determining whether \(G\) contains a subgraph that is isomorphic to \(H\) is a way to determine proximity is a method to compare graph similarities [6].

Graph kernels: Kernels have been popularized in the machine learning community as a powerful feature mapping tool, especially when combined with SVM classifiers. With graph structures, it is possible to define kernels that share same properties [32]. For implementation of several kernels, we refer to [28].

Example 1: Shortest-path kernel

\[ D(G) \] denotes the set of shortest distances between all node pairs in a graph \(G\). The shortest-path kernel value on two given graphs \(G_1\) and \(G_2\), is computed:

\[ k(G_1, G_2) = \sum_{sd_i \in D(G_1)} \sum_{sd_j \in D(G_2)} \mathbb{I}(sd_i = sd_j) \]  

Example 2: Pyramid match kernel: Pyramid match graph kernel uses a bag-of-vector representations of two given graphs. The idea of the algorithm is to map these vectors to multi-resolution histograms, and to compare these histograms with a weighted histogram intersection measure in order to find an approximate correspondence. For more details on this kernel, we refer to [24].

To compare query and neighborhood graphs of words, we selected graph kernels for two reasons. First, graph kernels are offer lots of options due to various kernel definitions. Second, we conjectured (based on datasets adapted for named entity linking) that subgraph isomorphism is condition too strong for named entity linking.

Given these definitions, our second method is obtained by adapting previous routine (eq. 6, and 7, algorithm 2) replacing the scoring function \(s\) by graph similarities into algorithm 3. The final routine (inference) for both methods is summed up in algorithm 4.

"Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources"
Algorithm 3 Graph of Words/Graph Kernels (GoW/GK) method

Require: Same inputs as algorithm 2
1: for \(i = 1\) to \(K\) do
2: Get neighbor/close nodes in the graph of \(e_i\)
3: For each neighbor \(n_j\), compute graph of word representation
4: Aggregate original entity filtering score and graph similarity scores between neighbors and the query graphs of words
5: end for
6: return \(\langle S^i \rangle_{1 \leq i \leq K}\)

Algorithm 4 SGM and GoW/GK named entity identification (Inference)

Require: Knowledge base \(B\) and its graph \(G_B\), mentions \((M_j)_{1 \leq i \leq M}\), scoring threshold \(K\), trained predictor \(F\)
1: for \(i = 1\) to \(M\) do
2: Use algorithm 1 on mention \(M_j\) and \(B\), return a list of \(K\) top ranked entities \((E^i_k)_{1 \leq k \leq K}\)
3: Use algorithm 2 or 3 using \(G_B\), on \(K\) entity candidates, return new score vectors
4: Evaluate \(\hat{F}\) on each vector score and use maximum a posteriori to infer estimated gold entity \(\hat{G}_i\)
5: end for
6: return \((\hat{G}_i)_{1 \leq i \leq M}\) (list of estimated gold entities)

3.4 Computational complexity

Our filtering algorithm time complexity is upper-bounded by \(|M||E| = ME\). SGM procedure (algorithm 2) time complexity is proportional to \(O(M|T|K)\), where \(|T|\) is the number of entity types considered, and \(K\) the remaining entity candidates. Similarly, \(G\) representing the kernel computational complexity, we can compute the complexity of our second method. In practice, \(G\) is proportional to the number of words in the query times the number of words in an entity description, which is reasonable since entity descriptions and queries can be considered as short texts (less than thousand of words). Therefore, the first method \((F+SGM)\) complexity is: \(O(M(E + K|T|))\), and the second method \((F+GoW/GK)\) complexity is: \(O(M(E + K|T|G))\). We report this in figure 9, along with some experimental computing times.

4 EXPERIMENTAL SETUP AND EVALUATION

4.1 Configuration

Datasets: Our datasets generated from CoNLL and NIST TAC-KBP 2009-2010, contain for each query its gold entity id and type.

Figure 6 gives the number of samples for each dataset; more details are available on their respective official websites.\(^2\), \(^3\).

NIL-detection: A nil mention is mention that has no entity identified in the knowledge base. As mentioned in the first part, we did not include in our work the problem of NIL-detection (de-tection of entities out of the knowledge base). Following [9, 10], we removed NIL entities from the datasets. Therefore, performance comparison with other solutions (cf figure 7) has to take in account this feature.

Implementation: Source code is available on demand.

Entity types, ontology: As discussed throughout the paper:
- Our methods rely on a fined-grained classification of entities in the knowledge base. To generate fine-grained entity type inside the knowledge base, we crossed DBPedia with NIST TAC-KBP knowledge base using entity Wikipedia titles, and CoNLL with the 2016 Wikipedia Dump.
- We did not include fined-grained entity recognition on the queries: we suppose this given as input in the data.

Graph kernels & regressors:
- We report here results (figure 7, 8, 9) with the pyramid match graph kernel, for its low complexity among standard kernels [24]. We tried different graph kernels for our second method, including Shortest-path kernel, Weisfeiler-Lehman Kernel, and results were similar.
- We tried several standard classifiers: regression trees, Support vector machines, and logistic regression. The results reported are obtained with logistic regression.

Performance metrics: A named entity linking system works as a search engine system, where we suppose there is only one relevant item (i.e. the gold entity). Since we don’t include NIL detection in our work, the most natural performance measure of such a system is to evaluate the presence of the associated gold entity in the top K results. Averaging this quantity over the samples gives the precision or recall at Top K (same quantity since the gold entity is either in or out the top K). The main motivation to display results at K comes from industrial applications, where the top K entities would be human annotated to identify the good entity (examples: database matching, company identification). Therefore, in this context, Top-K precision is important. Finally, comparing a NEL scorer and NEL classifier using accuracy gives misleading results because of unbalanced classes.

Standard deviation, statistical significance: We included standard deviation of the accuracy, but could not include p-significance of our method, due to the difficulty to reproduce other baselines experiments, namely:
- Source code is not publicly available
- In case of deep learning methods, specific embeddings are not released

| Dataset          | Non-NIL mentions | Total |
|------------------|------------------|-------|
|                  | PER   | ORG   | GPE   |
| TAC09 (Test)     | 255   | 1013  | 407   | 1675 |
| TAC10 (Test)     | 213   | 304   | 503   | 1020 |
| CONLL (Test)     | -     | -     | -     | 4379 |
| TAC10 (Train)    | 335   | 335   | 404   | 1074 |
| TAC14 (Train)    | 1461  | 767   | 1313  | 3541 |
| CONLL (Train + Valid) | -     | -     | -     | 22516 |
| Method       | NIL detection | Fine-grained entity types | Training samples nb. | Average P@1 (Accuracy) ± std in % |
|--------------|---------------|---------------------------|----------------------|-----------------------------------|
| Ganea, 2016 [9] | PGM          | No                        | Not required         | TAC09: 87.39, TAC10: 92.22       |
| Ganea, 2017 [10] | PGM & D.L    | No                        | Not required         | /                                 |
| Sun, 2015 [30]  | D.L           | No                        | Not required         | 82.26, 83.92                      |
| Yamada, 2016 [33] | D.L         | No                        | Not required         | 85.2, 93.1                        |
| Yamada, 2017 [34] | D.L           | No                        | Not required         | 87.7, 94.3                        |
| Globerson, 2016 [11] | D.L         | Yes                       | Not required         | 87.2, 92.7                        |
| Sil, 2017 [29]   | D.L           | Not Detailed†             | Not required         | 87.4, 93.0                        |
| Raiman, 2018 [26] | D.L           | Not Detailed†             | Required             | 90.85, 94.87                      |
| Guo, 2011 [14]   | Graph-based   | Yes                       | Not required         | 84.89, 82.40                      |
| Han, 2011 [17]   | Graph-based   | Yes                       | Not required         | /                                  |
| Hoffart, 2011 [18] | Graph-based | Yes                       | Not required         | /                                  |
| Usbeck, 2014 [31] | Graph-based | Yes                       | Not required         | /                                  |
| F+SGM          | Graph-based   | No                        | Required             | 94.58 ± 0.05, 93.66 ± 0.06, 92.70 ± 0.07 |
| F+GoW/GoK      | Graph-based   | No                        | Required             | 93.67 ± 0.06, 94.70 ± 0.05, 93.56 ± 0.06 |

Figure 7: Comparison of our approach to state-of-the-art methods with $K = 7$. For F+SGM, hyperparameter $\phi$ is obtained with cross-validation. †Not detailed if NIL-detection method is not mentioned or explained. D.L stands for deep learning and PGM for probabilistic graphical model.

- Filtering method is not detailed
- Routine for tuning parameters is not explicit

4.2 Results, comments and comparison

**Results**: We compare our methods with most performing baselines. Figure 7 sums up our experimental results (averaged P@1 is also referred as accuracy [30]). Our method performs better than any existing graph-based methods. It outperforms all existing methods on two NIST TAC09 and TAC10, and is competitive with state-of-the-arts methods on CoNLL/AIDA.

**Impact of parameter K**: We report impact of parameter K on final average P@1. Results are shown on figure 8. The curves show an experimental trade-off between exploration and a strict candidate filtering. Low values of $K$ don’t allow enough entity exploration and cause a drop in precision. On the contrary, high values of $K$ yields too many entity candidates. Results are similar for $5 \leq K \leq 10$.

**Scalability**: Figure 9 sums up asymptotic computational complexities and experimental computing time on a sub-instance of CoNLL/AIDA (1000 queries, 2.8 million entities). These times are only indicative, since there is room for improvement, especially due to the choice of the language (here, Python) and code optimization.

**Comparison: advantages and limitations**: Our methods yields remarkable precision P@1 on TAC09 dataset, CoNLL/AIDA and TAC10 datasets. For features extraction, the number of parameter that need to be tuned is reasonable, namely:

- F+SGM: the type mapping function $\phi$ (obtained with cross validation) and $K$
- F+GoW/GoK: graph of words window size, and $K$

For named entity discovery, our methodology has two limitations. First, we did not include NIL-detection. Second, our filtering method depends on fine-grained classification of named entities (example: GPE: Cities, Countries, ORG: Company, SoccerClub, PER: Actor, Singer, Politician, ...). We supposed such classification was available in the query, whereas progress still has to be made to tag named entities with fine-grained classification [21].
| Method     | Computational complexity | Experimental time (mn.) |
|------------|--------------------------|-------------------------|
|            |                          | Setup 1 | Setup 2 | K = 5 | K = 20 | K = 5 | K = 20 |
| F          | O(ME)                   | 620     | 620     | 44    | 44     |
| SGM        | O(MK[T])                | 240     | 290     | 14    | 16     |
| GoW/GK     | O(MK[T][G])             | 2642    | 306     | 16    | 20     |

Figure 9: Scalable entity identification. Computing times rounded to the minute. $M = 1000$, $E = 2.8 \times 10^6$, $G \leq 200$, $T = 250$. Setup 1: Single CPU with 32Gb Ram, Intel(R) Xeon(R) CPU E5-2407 4-cores 2.40GHz. Setup 2: Distributed cluster with variety of 20 CPU processors equivalent to setup 1. (using Spark/Hadoop technology)

5 CONCLUSION

In this paper, we proposed a new methodology concerning the problem of named entity linking. Capitalizing on experimental factors of entity mis-identification, we first proposed a filtering algorithm based on standard information retrieval techniques. Then, each entity candidate is matched with new features built on a knowledge subgraph centered on their corresponding node. Our methods perform individual linking: mentions are considered separately. Even though we did not include NLI detection or fine-grained entity recognition, we have shown empirically that our graph-based named entity identification outperforms state-of-the-art methods on two datasets and is competitive on one dataset.

We have also show that our filtering and graph-mining features extraction scales well: their computational complexity is linear with respect to the numbers of queries and entities, and they have good experimental computing time for short text documents.

There are some advantages of our method over deep learning approaches. First, entity features are interpretable. Second, our linking system is relatively easy to implement in a real system, with relatively few hyperparameters, especially for the second method using graph kernels. Last but not least, it does not require lots of data to reach good experimental performance. Indeed, only a few thousands of training samples were used to reach these results.

We hope this work will serve as a baseline for named entity linking when fine-grained entity ontology is available. This work also invites us to complete it with graph based named entity type classification. Moreover, we could potentially improve performance with careful attention given to a new graph kernel for named entity linking. We leave these ideas for future work.

REFERENCES

[1] Ayman Alhelbawy and Robert Gaizauskas. 2013. Named entity disambiguation using lms. In Proceedings of the 2013 IEEE/ACM International Joint Conference on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 03 IEEE Computer Society, 159–162.
[2] Ayman Alhelbawy and Robert J Gaizauskas. 2014. Graph Ranking for Collective Named Entity Disambiguation. In ACL (2) 75–80.
[3] A. Apostolico and C. Guerra. 1987. The longest common subsequence problem revisited. Algoriathmaica 2, 1 (01 Nov 1987), 315–336. https://doi.org/10.1007/BF01840365
[4] Rui Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and space-efficient entity linking for queries. In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, ACM, 179–188.
[5] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. Journal of machine Learning research 3, Jan (2003), 993–1022.
[6] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. A (sub graph isomorphism algorithm for matching large graphs. IEEE transactions on pattern analysis and machine intelligence 26, 10 (2004), 1367–1372.
[7] Maud Ehrmann, Leonida Della Rocca, Ralf Steinberger, and Hristo Tanev. 2013. Acronym recognition and processing in 22 languages. arXiv preprint arXiv:1309.6185 (2013).
[8] Paolo Ferragina and Ugo P Sciaila. 2010. 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, ACM, 1625–1628.
[9] Octavian-Eugen Ganea, Marina Ganea, Aurelion Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In Proceedings of the 25th International Conference on World Wide Web. Interna- tional World Wide Web Conferences Steering Committee, 927–938.
[10] Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambig- uation with local neural attention. arXiv preprint arXiv:1704.04920 (2017).
[11] Amir Gloeberson, Nevena Lazic, Soumen Chakrabarti, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. 621–631.
[12] David E. Goldberg. 1989. Genetic Algorithms in Search, Optimization and Machine Learning (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[13] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722 (2014).
[14] Yuhang Guo, Wanxiang Che, Ting Liu, and Sheng Li. 2011. A graph-based method for entity linking. In Proceedings of 5th International Joint Conference on Natural Language Processing, 1010–1018.
[15] Dan Gusfield. 1997. Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge university press.
[16] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R Curran. 2013. Evaluating entity linking with Wikipedia. Artificial intelligence 194 (2013), 130–150.
[17] Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, ACM, 765–774.
[18] Johannes Hoffart, Mohamed Amor Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Finkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 782–792.
[19] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In International Conference on Machine Learning, 957–966.
[20] Yang Li, Chi Wang, Jianqiu Han, Jiawei Han, Dan Roth, and Xifeng Yan. 2012. Fine-Grained Entity Recognition.. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’13). ACM, New York, NY, USA, 1070–1078. https://doi.org/10.1145/2457575.2457681
[21] Xiao Ling and Daniel S Weld. 2012. Fine-Grained Entity Recognition. In AAAI. Vol. 12. 94–100.
[22] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classiﬁcation. Linguisticsticae Investigationes 30, 1 (2007), 3–26.
[23] Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2016. J-NERD: joint named entity recognition and disambiguation with rich linguistic features. Transactions of the Association for Computational Linguistics 4 (2016), 215–229.
[24] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Matching Node Embeddings for Graph Similarity. In Proceedings of the 31st AAAI Conference on Artiﬁcial Intelligence, 2429–2435.
[25] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 1532–1543.
[26] Jonathan Raiman and Olivier Raiman. 2018. DeepType: Multilingual Entity Linking by Neural Type System Evolution. arXiv preprint arXiv:1802.01021 (2018).
[27] François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and TW-IDF: New Approach to Ad Hoc IR. In Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (CIKM ’13). ACM, New York, NY, USA, 59–68. https://doi.org/10.1145/2505515.2506571
[28] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skiani, and Michalis Vazirgiannis. 2018. GraKeL: A Graph Kernel Library in Python. arXiv preprint arXiv:1806.02193 (2018).
[29] Aqvilir Su, Gourab Kundu, Radu Florusan, and Wael Hamza. 2017. Neural cross-lingual entity linking. arXiv preprint arXiv:1712.01813 (2017).
[30] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation. In IJCAI. 1333–1339.
[31] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Athaide Coelho, Sören Auer, and Andreas Both. 2014. AGDISTIS-graph-based disambiguation of named entities using linked data. In International Semantic Web Conference. Springer, 457–471.

[32] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. Journal of Machine Learning Research 11, Apr (2010), 1201–1242.

[33] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. arXiv preprint arXiv:1601.03343 (2016).

[34] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning Distributed Representations of Texts and Entities from Knowledge Base. arXiv preprint arXiv:1705.02494 (2017).

[35] Wei Zhang, Yan Chuan Sim, Jian Su, and Chew Lim Tan. 2011. Entity linking with effective acronym expansion, instance selection, and topic modeling. In IJCAI. Vol. 2011. 1909–1914.