Collective disambiguation in entity linking based on topic coherence in semantic graphs

Efrén Rama-Maneiro[a,*], Juan C. Vidal[a], Manuel Lama[a]

[a]*Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)*
Universidade de Santiago de Compostela, 15782 Santiago de Compostela, SPAIN

Abstract

Entity Linking (EL) consists of determinating the entities that best represent the mentions in a document. Mentions can be very ambiguous and can refer to different entities in different contexts.

In this paper, we present ABACO, a semantic annotation system for Entity Linking (EL) which addresses name ambiguity assuming that the entity that annotates a mention should be coherent with the main topics of the document. ABACO extracts a sub-graph from a knowledge base which interconnects all the candidate entities to annotate each mention in the document. Candidate entities are scored according to their degree of centrality in the knowledge graph and their textual similarity with the topics of the document, and worst candidates are pruned from the sub-graph.

The approach has been validated with 13 datasets and compared with other 11 annotation systems using the GERBIL platform. Results show that ABACO outperforms the other systems for medium/large documents.

Keywords: Entity Linking, semantic annotation, topic coherence, Named Entity Disambiguation

1. Introduction

Annotations are metadata, such as phrases, comments, labels, or other types of external remarks that can be attached to a document or to a part of a document to give additional information about an existing piece of data [1]. Semantic annotation extends this concept and goes one step further to reduce the gap between natural language and its computational representation, trying to match terms of a document with its semantic representation in a knowledge base (KB) [2]. The main complexity of this process is associating an entity of a knowledge base to a specific term in the document, since the same term can be ambiguous and have multiple meanings. For example, the term "Oreo" can refer to a type of cookie, a fish, or a version of the Android operating system, among other entities. Thus, finding the correct individual in the knowledge base is essential to provide an accurate annotation. This problem, known as Entity Linking (EL) [3], has become more relevant in semantic annotation with the emergence of large repositories of Linked Data (LD), such as DBpedia [4] or Freebase [5].

The key idea of EL is to identify the relevant terms (also called *mentions*) in the input text and link them to an individual (also called *entity* or, in this paper, *nodes*). This process usually involves three main steps: *(i)* parse the text to find candidate mentions to be linked, *(ii)* generate a set of candidate entities for each mention, and *(iii)* perform a disambiguation step to find the best candidate entity for each mention. The disambiguation step may be either *local*, that is, the disambiguation is performed taking into consideration each mention separately or *global*, that is, disambiguating the set mentions and their corresponding candidate entities conjointly.

Name ambiguity is usually solved by focusing on the mention context compatibility, on the document topic coherence, or both. *Context compatibility* assumes that the entity of a mention is reflected by its context, usually the surrounding words. For example, the mention "Oreo" in Figure 1 will be identified as a version of the Android operating system and thus rule out other candidates such as the *Oreo* cookie. On the other hand, *topic coherence* assumes that the entity of a mention should be coherent with the main document topics. For example, the mention "Alphabet" in Figure 1 should be linked to the entity *Alphabet Inc.* since the main topic of the document is the release of a new software version created by *Google* (a subsidiary).

*Corresponding author
Email addresses: efren.rama.maneiro@usc.es (Efrén Rama-Maneiro), juan.vidal@usc.es (Juan C. Vidal), manuel.lama@usc.es (Manuel Lama)

Early EL approaches used Wikipedia as its main knowledge base (KB) [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. However, the development of semantic repositories, such as DBpedia, changed this scenario and nowadays most approaches take advantage of the semantic features of knowledge bases to improve EL [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26] or combine DBpedia with other KBs such as Wikipedia, WordNet [27], or YAGO [28] (Yet Another Great Ontology) [29, 30, 31, 32, 33, 34, 35]. Furthermore, the recent increased popularity of deep learning techniques has propitiated the appearance of a set of approaches that use deep neural networks [36, 37, 38, 39, 40]

In this paper, we present ABACO[1,2,3] (Annotation BAsed on COherence), a novel EL approach that implements a method based on both collective disambiguation and document topic coherence. ABACO is meant to annotate medium to large documents since collective disambiguation does not work well with texts with a small number of mentions or with texts without a thematically homogeneous text. For example, the terms "Oreo", "Nougat", and "Donut" may end up mapped to confectionery-related entities, or three versions of the Android operating system, if the text is short and only contains unrelated or weakly related topics.

ABACO differs from state of the art in how it addresses collective disambiguation. Its objective is extracting the sub-graph from the DBpedia that best describes the topics of the document. For this purpose, it first creates a graph that contains all entities that are a candidate to annotate each mention and then looks for paths to interconnect these entities. Although some recent papers also solve collective disambiguation through graphs [14, 33, 34, 35, 38], in most of the cases it is based on selected features of Wikipedia or DBpedia. However, our approach is more suited for DBpedia since it provides a relation granularity that allows building precise subgraphs when the ontology is explored to a greater depth.

Candidate entities are scored *(i)* according to its degree of centrality (a node is relevant if many other nodes link to it, or if it links to many other nodes). However, and contrary to other approaches, we also score the entity *(ii)* by its semantic similarity with the document topics. Specifically, we make use of the full Wikipedia page of an entity as its semantic description.

[1]A web interface to the annotator is available at `https://tec.citius.usc.es/abaco/`

[2]REST services of the annotator are accessible at `https://tec.citius.usc.es/abacows/`

[3]The NIF webservice used for evaluation is accessible at `http://tec.citius.usc.es/gerbilendpoint/`

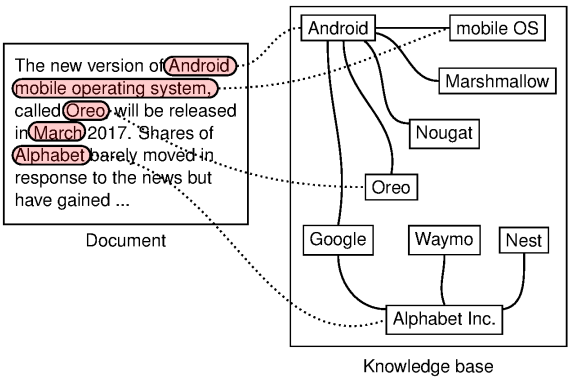

Figure 1: An example of Entity Linking

Moreover, ABACO has been designed to deal efficiently with massive disambiguation graphs. Even though other state-of-the-art approaches use disambiguation graphs [12, 14, 8, 41, 29, 33, 30, 42, 35, 50, 51, 52], the main improvement of ABACO is its ability of exploring the KB to a higher degree while mitigating the combinatorial explosion that involves exploring the KB. ABACO uses more relation types whilst discerning the inherent structure of the KB, mainly by filtering relations with less information, category relations, from relations that provide more information. ABACO can interconnect entities at any distance, although in the experiments, we have configured a maximum of 8 nodes of distance since that depth is sufficient to capture most of the significant relations between entities. The disambiguation depth is a crucial feature since the high granularity of the DBpedia categories promotes that entities can be at a considerable distance from each other. Therefore, methods that only explore a few relations and do not deepen in the DBpedia structure are unable to connect most entities and result in graphs containing isolated entities, making the disambiguation more difficult. Moreover, ABACO uses a bidirectional strategy and an indexation of DBpedia paths that alleviates the combinatorial explosion that involves exploring the DBpedia for building disambiguation subgraphs. Furthermore, the filtering of entities using the context compatibility between a mention and a candidate entity allows building graphs that are more coherent with the text to annotate and also reduces the search space of paths between candidate entities.

Summarizing, the main contributions of our graph-based disambiguation approach are as follows:

- A novel candidate generation algorithm that performs a precise early filtering of candidate entities. This limits the set of candidate entities *per* men-

tion thus reducing the time needed to build a disambiguation subgraph and enhances the coherence of the final built subgraph.

- A bidirectional strategy for building disambiguation graphs that drastically reduces the number of nodes needed to explore to build the graph and allows exploring the KB to depths higher than other state-of-the-art works.
- A mechanism for indexing KB paths that further reduces the time needed for building a disambiguation subgraph.

Finally, our approach has been validated with 11 other entity annotation systems, using the baselines defined in GERBIL [43] to compare annotation systems using 13 different datasets. This validation shows that ABACO achieves the highest mean F1 score of the tested datasets. Our approach performs consistently over a wide variety of texts, does not need any parameter tuning nor training and uses classical information retrieval techniques, showing that these techniques can have similar or even better performance than probabilistic or machine learning-based approaches.

The rest of the paper is organized as follows. Section 2 defines the Entity Linking problem approached in this paper. Section 3 analyses recent approaches to performing EL. Section 4 describes our graph-based annotation approach. Section 5 details the baselines for the evaluation of the entity annotation systems. Section 6 summarizes the results obtained by the different approaches. Finally, Section 7 points out the conclusions and future work.

2. Problem formulation

The main task in EL is to map a set of mentions $M = \{m_1, ..., m_n\}$ to a set of unambiguous candidate entities $C = \{c_1, ..., c_n\}$ of a *knowledge base* (KB) K.

Definition 1. *A knowledge base K is a directed graph $G_K = (V, R)$ where V is the set of nodes of K and R is a set of relations of K. Each node $v_i \in V$ of K represents an entity of the knowledge graph K. Each relation $r_i \in R$ of K is a tuple (v_i, e, v_j) where $v_i, v_j \in V$ and e is a property of the pair of entities v_i and v_j.*

Definition 2. *The EL problem can be modeled as a maximization problem as follows:*

$$\lambda^*_{m_i} = \underset{c_i \in C}{\operatorname{argmax}}[\Psi(m_i, c_i) \cdot (1 + \Phi(c_i, \Gamma))] \qquad (1)$$

where:

- *m_i is the current mention from the set of mentions target of the annotation problem M, i.e, $m_i \in M$.*
- *$\lambda^*_{m_i}$ is an assignment between a mention m_i and a single entity c_i, i.e, the solution to the EL problem for a single mention.*
- *C is the set of candidate entities that could be a solution for the assignment $\lambda^*_{m_i}$.*
- *Ψ is a local score function that calculates the context compatibility between a mention and a given candidate entity.*
- *Φ is an entity coherence function that calculates the topic coherence between a candidate entity c_i and a subset of a knowledge base K, Γ, that captures the relationships between all candidate entities for the set of mentions M (collective disambiguation). Thus, $\Gamma \subseteq K$.*

In [7] is shown that this maximization problem is NP-hard so approximate solutions are required. In this paper, we will define Ψ and Φ functions to approximately solve the EL problem. This problem is called *Named Entity Disambiguation* (also called *D2KB*), and it will be the focus of this paper. *D2KB* focuses on linking each mention of an already given set of mentions M to an entity in the knowledge base or return NIL for that mention if no entity can be found.

Thus, we tackle the full problem in which the entity mapping is $c_i \rightarrow K \cup \{NIL\}$, unlike other state-of-the-art approximations such as [44, 37, 36].

Note that this formulation is different from the one in [13] since the score is not a sum of the local score function and the entity coherence function, but a multiplication. This reformulation allows us to compensate for entities that are very popular in a KB but have incompatible contexts, or entities that are isolated in the knowledge graph but have a compatible context.

Summarizing, the objective of this paper is threefold: *(i)* devise an algorithm that finds a set C that contains the correct candidate entity c_i for a given mention, *(ii)* find a Γ that allows performing an efficient *collective disambiguation* of all mentions M, and *(iii)* define the Ψ and Φ functions to find the correct mapping $\lambda^*_{m_i}$ for a mention m_i.

3. Related Work

There exists a wide range of approaches regarding perform Entity Linking, whether they use Wikipedia or other KBs like DBpedia or YAGO. These approaches

can be classified into three broad categories: probabilistic approaches, graph-based approaches, and unsupervised learning-based approaches.

3.1. Probabilistic approaches

Probabilistic approaches try to find the probability of an entity *c* to be linked with a mention *m*.

The *Illinois Wikifier* annotation system is presented in [13]. Wikifier extends semantic relatedness measures between Wikipedia titles to disambiguate entities using document topic coherence. An anchor-title index is used to compute the fraction of times a title is a target page of another anchor and the fraction of all Wikipedia articles that link to this title, in addition to other local features such as a TF-IDF representation of a 100 token window around a mention. Every feature has a coefficient that is learned using a Support Vector Machine (SVM) with training data from Wikipedia.

In [10] authors propose a semantic relatedness score to measure how much two words are related by encompassing a set of features (title words, frequency, categories, and outgoing links) extracted from Wikipedia articles. All entities, their features, and relations are transformed into a graph, and a random walk is then applied to combine the effects of features and derive the relatedness score. Following a similar approach, *TagMe 2* annotation system is presented in [11]. TagMe scores candidate entities combining a correlation distance, obtained from a simple graph representation of Wikipedia, and the probability to link the entity to a mention.

FOX [45] is an ensemble-learning based framework that uses four different NLP algorithms (Stanford Named Entity Recognizer, Illinois Named Entity Tagger, Ottawa Baseline Information Extraction, and Apache OpenNLP Name Finder) to search the best ensemble classifier, allowing to overcome the weakness that each NLP algorithm has individually. Another approach based on ensemble of methods is *NERD-ML* [46] where a machine learning classifier is trained using *(i)* multiple NER extractors (such as AlchemyAPI, DBpedia Spotlight, Extractiv, Lupedia), *(ii)* a feature extraction from mentions to annotate such as prefixes, suffixes or whether the mentions are at the beginning or end of the document, and the *(iii)* Stanford NER system trained on the MSM training dataset. These components are then assembled on a single machine learning algorithm.

KEA [47] is based on a fine-granular context model that uses multiple text sources and automated multimedia analysis. Ambiguity is solved in three steps: *(i)* an n-gram analysis algorithm, *(ii)* searching for all potential DBpedia candidate entities for each n-gram, and *(iii)* selecting the best entity according to the probability calculated using a score cascade. *PBOH* [44] is a pure entity disambiguation system that uses a probabilistic graphical model using pairwise Markov Random Fields to perform collective entity linking. This model is based on contextual word-entity statistics extracted from the English Wikipedia corpus, also considering the anchor text of the hyperlinks. Another approach that uses Conditional Random Fields is *FREME NER* [48], an e-entity service published by the EU project FREME, which makes a selection of candidate entities based on a sense of commonness within a KB.

More recently, some approaches try to perform entity disambiguation using deep learning techniques. These approaches extract entity embeddings that are used to train a deep neural network which acts as a classifier that matches entities to mentions. *Yamada et al.* [36] create a convolutional neural network with embeddings of the positions of the context words to determine the distance between context words and mentions. Furthermore, they use a low-rank neural tensor network to model the semantic composition between context and mention. *DeepEd* [37] uses deep learning to perform joint document-level named entity disambiguation. They extract entity embeddings, use neural attention mechanisms over local context windows which allows selecting words that are informative for the disambiguation and perform a differentiable joint inference stage which executes the disambiguation step collectively. *DeepType* [40] proposes a 2-step algorithm that *(i)* performs a heuristic search or stochastic optimization over discrete variables that define a type system informed by an Oracle and a learnability heuristic *(ii)* trains a neural network to build a classifier using this type system. Thus, they restrict the output of the neural network to conform to the type symbolic system previously trained.

3.2. Graph-based approaches

Graph-based approaches build a subgraph from a KB that interconnects candidates associated with mentions. In [12], the authors exploit the Wikipedia link structure to model the knowledge as a graph where articles and categories are nodes. Local subgraphs (up to 2 levels deep) containing the candidate Wikipedia articles for all entity mentions in the document are scored using centrality measures (degree centrality and PageRank [49]). In [14], a graph-based representation which can capture both the local mention-to-entity compatibility and the global interdependence between entities is presented. Mention-to-entity compatibility measures the likelihood between the context of the mention and the

entity. Global interdependence between entities extends the semantic relatedness proposed in [8]. A third feature, called evidence, is used to measure the importance of the entity in the context. *OpenTapioca* [41] uses a similar approach as [14] using as a local compatibility features the popularity of an entity (measured by means such as its Pagerank or site links) and the probability of a mention occurring in a text (measured by a unigram model) and as a semantic similarity measures a weighted graph over the Wikidata knowledge graph. Then they train a support vector classifier over the combined Markov chain of these features.

AIDA annotation system [29] is based on the YAGO2 KB. It also disambiguates focusing on context similarity but using an undirected graph that has two kinds of edges: *(i)* mention-entity edges which are weighted using a similarity or/and a popularity measure and, *(ii)* entity-entity edges weighted using Wikipedia-link overlap, type distance or both. Their objective is to find a graph that contains all mention nodes and exactly one mention-entity edge *per* mention. This graph is computed by maximizing the concept of *density* of the graph, which is the minimum sum of the weights of its incident edges.

Babelfy [33] annotation system combines lexicographic knowledge used in word sense disambiguation for tackling EL. This approach uses *BabelNet* [30] (a KB created from Wikipedia and WordNet) to identify the candidate entities and creates a graph-based semantic interpretation of the whole text by linking candidate meanings using previously-computed semantic signatures. A dense subgraph of this representation is extracted to select the best candidate for each text fragment. *AGDISTIS/MAG* [42] is an evolution of the AGDISTIS [35] annotation system which is adapted to tackle multilingual problems. It generates a disambiguation graph from candidate entities and uses a bread-first search to retrieve the context information from the KB. Then, Hypertext-Induced Topic Search (HITS) algorithm scores the entities. Cetoli et al. [50] study the possibilities of incorporating topological information from a Knowledge Graph into a neural network using various techniques (LSTMs, Graph Convolutional Neural Networks, etc.) and representations. They rely on the Wikidata Knowledge Graph, for building the entities graph, and in Glove embeddings, for creating vector representations from the text, nodes, and edges. Khalife et al. [51] extract a feature vector directly from the knowledge graph to train a classifier to rank candidate entities. This feature vector is a concatenation of the similarity scores between a given query and the candidate entity neighbors. The similarity is measured using a graph kernel over a graph of words representation of the entity and the query. GEEK [52] generates a knowledge graph using the Google Knowledge Graph that interconnects a set of candidate entities. In this case, entities are weighted using three measures: the score from the query to the Google Knowledge Graph API, the document similarity between the short descriptions of entities and the input text, and the Wikipedia Link-based Measure between two entities.

Note that not every graph-based approach needs training since, like ABACO, they can calculate context compatibility and topic coherence using static measures such as string similarity or PageRank. This contrasts with probabilistic approaches that use supervised learning methods such as Support Vector Machines [13], Markov Random Fields [44] or Deep Learning techniques [36, 37, 40].

3.3. Unsupervised learning-based approaches

Unsupervised approaches do not need a manually labeled corpus to perform the entity disambiguation. *DBpediaSpotlight* [18] is one of these approaches. It represents entities in a vector space model where each DBpedia entity is a point in a multidimensional space of words. Entities are disambiguated comparing the mention's context, e.g., surrounding paragraph, and the entity, a document containing the aggregation of all paragraphs mentioning that entity in Wikipedia, using cosine similarity.

Another example is [9], authors considered a different relatedness feature that matches common nouns in the mention's context against Wikipedia article names, and then used hierarchical agglomerative clustering to select the best candidate entity. In [7], a title relatedness is proposed, based on the overlap in categories and incoming links. They build a vector for each candidate entity by identifying the set of entity references appearing in the text. Then, a maximization process is performed between the candidate entity and the entity mention and between the categories associated with candidate entities.

3.4. Limitations of current approaches

The previously described methods have some limitations. On the one hand, machine learning-based approaches [13, 45, 46, 48] perform the disambiguation of each mention of the document separately, so the semantic coherence between mapped entities is reduced. On the other hand, probabilistic and unsupervised approaches [44, 47, 11, 18, 9, 7] often use Wikipedia statistics regardless of its relation with the

document topic, i.e., some statistics, such as incoming link count or the Wikipedia article hyperlinks do not take into account whether the link that points to the article or the article hyperlinks are directly related to the other entities mapped in the document. Furthermore, most of the annotators in this category, such as PBOH [44], only consider simple statistics such as entity-mention cooccurrence. Regarding deep learning approaches [36, 37, 40, 50], they require an expensive training procedure, often using GPUs, due to the complexity of using a deep learning model. Furthermore, these approaches use entity embeddings in which its annotation accuracy depends on the corpus used to train those embeddings and, thus, making the neural model biased towards specific datasets.

Finally, graph-based approaches [12, 14, 8, 29, 33, 30, 42, 35, 50, 51, 52] often explore the KB to a small degree due to the space of possible paths growing combinatorially. For instance, AGDISTIS/MAG is limited to up to two levels of relations to disambiguate due to this combinatorial explosion. Therefore, the disambiguation of a candidate entity is hard since the graph only contains a few links between entities or isolates entities from each other.

Note that other approaches filter entities based on different criteria such as the types of the entities [35], since they only disambiguate named entities; the distance from the mentions to the entity nodes in their mention-entity graph [29]; the similarity between the labels of the entities and the mention [42, 51]; dense subgraphs [33, 42]. Finally, there are approaches that do not apply any kind of filtering [12, 14, 41, 50, 52].

4. Graph-based collective disambiguation approach

In our approach, EL is based on the assumption that document topic coherence should be preserved from the set of mentions to the set of entities used to annotate the document. ABACO tries to maintain coherence using a semantic model extracted from the KB, thus approaching *disambiguation globally*, and not individually for each mention. Note that, as we try to find an approximate solution for equation 1, we define the BM25 ranking function as the Ψ function and a personalized centrality measure as the Φ function. It is vital to highlight that the Φ function domain is not the entire KB since this is not tractable from a practical point of view, but this domain is a sub-graph that connects all the entity candidates. This property is one of the main differences between ABACO and the current graph-based approaches.

Figure 2 shows the process followed by ABACO to deal with the entity linking problem. This process has four main steps that must be executed sequentially. In the first step, the terms that are present in the text are extracted, thus configuring the *context* of the given document. We gather every term present in the text since the given mentions do not have to be the same as the extracted mentions in this step. In the second step, for each given mention, we generate a rank of the most probable candidate entities that could be linked to each mention target of the annotation process. To perform this ranking, the textual similarity is measured between the context of the document and each candidate entity. In the third step, the sub-graph associated with the generated entity candidates in the previous step is built using a bidirectional strategy. In the fourth and final step, the best candidate is selected by weighting the sub-graph using a centrality measure and performing the selection of the best candidate for each mention.

The relevance of an entity is defined as a combination of its centrality and its document textual similarity, calculated as follows:

$$R(v) = sim(W, Q_n) * (1 + C(v)) + \alpha \tag{2}$$

where:

- $R(v)$ denotes the relevance of a given node.
- $sim(W, Q_n)$ denotes the textual similarity of the Wikipedia page of an entity and the extracted context from the text.
- $C(v)$ is the centrality calculated for the node v.
- α is an adjustment factor.

Note that 1 must be added to the centrality to avoid canceling the relevance in case a node is not linked to any candidate, even though it could still be relevant as far as textual similarity is concerned.

Recalling from equation 1, in which we defined Ψ as the local score function and Φ as the entity coherence function, ABACO performs the assignment Ψ function as $sim(W, Q_n)$, that is, the textual similarity of the Wikipedia page for an entity and the extracted context is our local score function, and Φ as $C(v)$, that is, the centrality of a candidate entity is the topic coherence function.

In the following subsections, we will detail each one of the parts of this approach and how the different items of equation 2 are obtained.

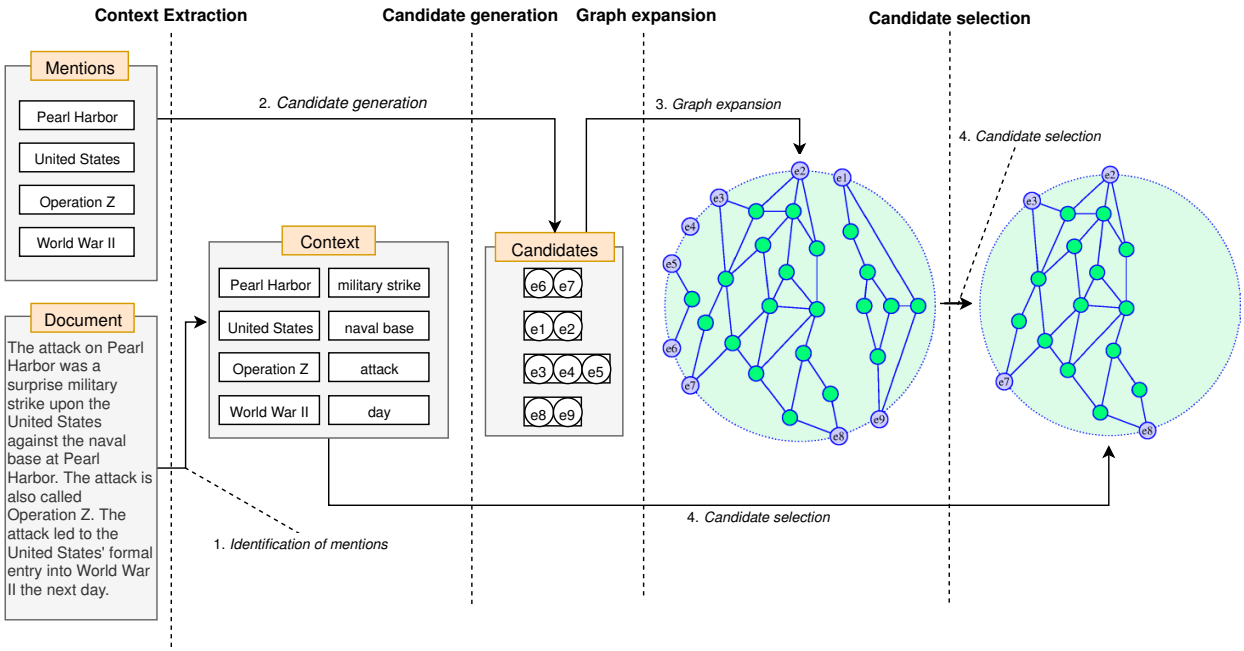

Figure 2: The 4-part annotation process. This process has as inputs the text document and mentions to annotate from that document. In the first step, context extraction, a set of terms is extracted from the text document (note that the mentions extracted here are not the same as the mentions given as input of the process). In the second step, candidate generation, a set of nodes (candidate entities) is generated for each mention given as a challenge to annotate. In the third step, graph expansion, every set of candidates is interconnected to build a graph that represents the semantic relations from the KB of the mentions to annotate (note that candidates of the same set, such as e_1 and e_2, are not interconnected). In the fourth step, candidate selection, the best candidate for each set of nodes is selected, taking into account the context extracted and the graph previously built.

4.1. Identification of mentions

Even though the disambiguation problem already gives the mentions of the text to annotate, often they are not enough to get the main topic of the document. Thus, it is imperative the detection of every possible mention in the given text to build a context which will be used to compute the semantic similarity between candidate entities and the text topic.

To perform this detection, we rely on an n-gram algorithm that extracts mentions which are either *(i)* a named entity, regardless of its existence on the DBpedia or *(ii)* the longest n-gram available in a given window that exists as a resource on the DBpedia.

Algorithm 1 shows this n-gram approach. Line 1 defines the maximum number of words, forward and backward from the word detected by the POS Tagger. The POS Tagger assigns to a sequence of tokens a Named Entity category [53]. In lines 2-3 a NER system is used to detect possible named entities from the text. Note that these mentions do not have a matching entity in the KB. In line 4 the POS Tagger is called to get the substantives from the text, which will be the base to search possible n-grams. In lines 5-7 an iteration is performed over every detected substantive and every combination of possible windows. In line 8 a possible n-gram is extracted directly from the text, including stopwords since some mentions can contain some stopwords in the middle of two substantives, such as `Bank of Spain`. If the n-gram exists in the KB, we stop trying combinations of windows and a new search is started from a new substantive. As a NER and POS Tagger system, ABACO relies on the Stanford CoreNLP [54] toolkit.

To check whether a mention is present in the DBpedia the following indexes are used:

- *Label index*. This index has been built from the `rdfs:label` property of every entity available in the DBpedia. Querying this index returns the main entity associated to the mention or/and its disambiguation page (if either of these two resources exist).

- *Synonym index*. This index has been built from the `dbo:wikiPageRedirects` of every entity available in the DBpedia. Querying this index allows getting the canonical name of an entity (its *redirect*) or its list of possible synonyms.

Algorithm 1: N-Gram detection algorithm

```
   Input: t - given text document
   Output: L list of mentions
1  const WINDOW ← (MAX_WINDOW - 1) / 2;
2  named_entities ← detectNER(t);
3  push(L, named_entities);
4  substantives ← posTag(t);
5  for name in substantives do
6      for back = WINDOW to 0 do
7          for forward = WINDOW to 0 do
8              ngram ← getNgram(name, text, back,
                 forward);
9              if existsInKB(ngram) then
10                 push(L, ngram);
11                 break;
12             end
13         end
14     end
15 end
16 return L
```

Thus, to check the existence of a mention in the DBpedia we first check whether the mention exists in the label index and, if it does not exist, we check if it is a redirect. This checking process is performed by trying different combinations of surface forms such as exact matching, heuristic plural removal, and lemmatization using Stanford NLP. Note that in this algorithm we are only interested in the existence of mentions that have an entry in the DBpedia and not in their possible relevance in the text, since in the D2KB problem, the mentions to link have all the same relevance, i.e., every mention must be considered to be linked to a DBpedia entity.

The complexity of Algorithm 1 is $\mathcal{O}(n)$, established by the loop in line 5. This loop is upper bounded by the number of substantives detected by the POS Tagger. The other nested loops in lines 6 and 7 have a $\mathcal{O}(1)$ complexity, bounded by the constant WINDOW.

The only parameter needed in Algorithm 1 is MAX_WINDOW. This parameter controls the size of the window of tokens extracted given a mention position in the text which is applied symetrically around the mention. Therefore, the value of MAX_WINDOW must be odd. The bigger the window is, the longer n-grams the algorithm is able to detect, but more time is spent doing checks in `for` loops in lines 6 and 7. In our experimentation, we found that a value of 5 for MAX_WINDOW has the best ratio precision-performance.

4.2. Candidate generation

After the detection of every mention in the document, ABACO identifies a set of candidate nodes susceptible to be the correct annotation for each given mention. This set of candidates is *coherent*, that is, the set of candidates is a selection of the best nodes found for a given mention that are as semantically close as possible. To perform this task, ABACO uses the indexes aforementioned in 4.1 plus the following indexes:

1. *Disambiguation index*. This index has been built from the `dbo:wikiPageRedirects` DBpedia property. Querying this index allows retrieving the list of possible entities contained in a specific disambiguation page.

2. *Wikipedia index*. This index has been built from the Wikipedia dump from the 01 of January of 2018. It allows retrieving the full Wikipedia page from a given DBpedia URI, and it is used to calculate the textual similarity between an extracted context and a possible candidate entity, represented by its corresponding Wikipedia URI.

3. *Type index*. This index has been built from the `rdfs:type` and allows getting the list of types associated with a DBpedia entity to perform an early filtering of non-relevant entities.

4. *Contextualization index*. This index aids the contextualization of every candidate entity associated with a mention of the text. It contains, among others, a list of demonyms created from a Wikipedia page[4], a list of prepositions that may indicate whether a mention may correspond to, a country or not, a list of country abbreviations[5] or a list of corporations abbreviations (such as Ltd. or Corp.).

Thus, the candidate generation algorithm has three main steps annotate each mention: *(i)* heuristically solve the coreferences in the text and generate a set of candidate entities from different kinds of surface forms of the mention, *(ii)* explore the disambiguation pages found in the previous step recursively in order to expand the candidate list as much as possible, and *(iii)* apply a type inference system to generate or delete some candidates and use the textual relations to prune candidates not related to the text. Algorithm 2 explains in detail these steps.

The inputs of this algorithm are: *(i)* the set of mentions to annotate, *(ii)* the full-text document, and *(iii)* context extracted from Algorithm 1.

[4] `https://en.wikipedia.org/wiki/List_of_adjectival_and_demonymic_forms_for_countries_and_nations`

[5] `https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations`

Algorithm 2: Candidate generation algorithm

```
   Input: L - list of mentions to link, t document text, c
          extracted context
   Output: Map hashtable of candidate entities for each
           mention to link
1  Map ← [,];
2  const MAX_DISAM_DEPTH ;
3  for mention in L do
4      candidateUris ← [];
5      mention ← solveCoreferences(t, mention);
6      exactUris ← searchSurfaceForms(mention);
7      push(candidateUris, exactUris);
8      for uri in exactUris do
9          if isDisamPage(uri) then
10             currentDisamDepth ← 1;
11             push(candidateUris, disambiguate(uri,
                 MAX_DISAM_DEPTH,
                 currentDisamDepth));
12         end
13     end
14     mainKeywords ← contextualizeText(t);
15     for uri in candidateUris do
16         if not isValidType(uri, mainKeywords) then
17             pop(uri, candidateUris);
18         end
19     end
20     if isCombinationPresent(mainKeywords, mention)
         then
21         push(generate(mainKeyWords, mention),
             candidateuris)
22     end
23     if isEmpty(candiateUris) then
24         push(fuzzySearch(mention), candidateUris)
25     end
26     push(map, {mention,
         weightTextRelations(candidateUris, c)});
27 end
28 return Map
29 procedure disambiguate(uri, MAX_DISAM_DEPTH,
     currentDisamDepth)
30     disamList ← [];
31     push(disamList, getCandidates(uri));
32     if MAX_DISAM_DEPTH = currentDisamDepth then
33         return [];
34     end
35     for disam_uri in disamList do
36         if isDisamPage(disam_uri) then
37             disambiguate(disam_uri,
                 MAX_DISAM_DEPTH, currentDisamDepth
                 + 1);
38         end
39     end
40     return disamList;
```

The only parameter of the algorithm is MAX_DISAM_DEPTH, which controls the depth to which the disambiguation pages in DBpedia are explored. The recursive exploration is needed since a disambiguation page could potentially point to more disambiguation pages. In our experimentation, we found that a value of MAX_DISAM_DEPTH of 2 is sufficient for retrieving a reasonable number of candidates from the disambiguation pages. Higher values of MAX_DISAM_DEPTH give repeated candidates, that is, given two disambiguation pages, A and B, A could point to B and B could point to A.

The output of the algorithm is a hashtable where keys are the mentions to be annotated, i.e, L, and the values of this hashtable are the most promising set of candidate nodes found for each mention. This hashtable is initialized empty (line 3). In line 4 the maximum exploration depth is defined for the DBpedia disambiguation pages, since a disambiguation page may point to more disambiguation pages. Then, in line 5 a coreference resolution system is applied to find a better surface form in the text to reduce the ambiguity of the mention to annotate.

The Stanford NLP Statistical Coreference System [55] supports this task and it is applied only if a coreference that contains the mention trying to annotate is found and that coreference exists in the DBpedia or is a named entity bigram. Note that the coreference system is not only used to find a better surface form.

The NER system to each promising found coreference, so the number of coreferences needed to explore is diminished. For example, if coreferences for the mention `Clinton` need to be solved and the first best coreference found is `President Bill Clinton`, `Bill Clinton` would still be extracted as a valid coreference.

In line 6 a surface form generation rule system is used to find the highest number of candidate nodes possible given a mention. For that, the contextualization index is queried to try as many promising surface form combinations as possible, in addition to applying other lemmatization techniques such as plural removal. Note that the queries performed to all indexes (except to the Wikipedia index) are all exact queries since partial match queries introduce much noise by generating candidates that are not directly correlated with the text.

In lines 8 to 13, the set of candidate nodes is enlarged by exploring the disambiguation pages found in the previous searching using the procedure `disambiguate`. This procedure queries the disambiguation index to retrieve the links contained in the disambiguation page that is being explored. This exploration is recursive since a disambiguation page may contain links to more disambiguation pages. This exploration continues until the threshold MAX_DEPTH is reached.

In lines 15 to 22 the type rule system is applied to prune some candidates based on its type or to generate some new candidates. First, an analysis is performed over the possible set of types for each term based on

its exact match occurrence in the type index and, then, they are classified in categories such as "locations" or "sports". Furthermore, a lookup for specific words that may give a hint about the possible topic of the document is performed, such as "film", "movie" or "album". Then, ABACO inspects every candidate entity generated for the current mention and based on the information gathered previously, discards URIs that have a low probability of being the entity associated with the mention. For example, if an entity of type `Film` is generated, but no other word in the text makes reference to films, that URI is discarded. Then, ABACO also checks for combinations of types that may indicate that a URI is prone to be the real candidate and thus, to be generated. For example, if ABACO detects by its type a sport in the text (e.g., `football`) and we are evaluating a mention that is a country (e.g., `Spain`) ABACO generates candidates by querying the type index and retrieving the set of nodes of the DBpedia that make reference to Spain and are `FootballTeams` (e.g., "Spain national football team", "Spain under 21 national football team" and so on). To reinforce the detection of countries, we use the contextualization index to check if the mention is, in fact, a country and avoid generating in that case.

In lines 23 to 25, a check is performed to assure that the previous generation steps were successful. If no candidate is found, a less restrictive search for candidates is performed. For that, the Wikipedia search API is queried since it takes into account not only the text searched but also statistics about the own Wikipedia pages such as popularity.

In line 26 the similarity between every candidate found and the context extracted in the previous process step is calculated to eliminate candidates that are not well aligned with the document context. The Wikipedia index is used, so the document context similarity problem is reduced to how well the Wikipedia page of the candidate aligns with the context extracted. In this paper, the Lucene BM25 ranking function is used to calculate the similarity between the Wikipedia page associated with the candidate entity and the context extracted from the text. In particular, the similarity is defined as follows:

$$sim(W, Q_n) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, W) \cdot (k_1 + 1)}{f(q_i, W) + k_1 \cdot (1 - b + b \cdot \frac{|W|}{avgl})} \quad (3)$$

where:

- $sim(W, Q_n)$ is the similarity between the Wikipedia page of the candidate entity (W) and the set of mentions extracted from the context (Q_n). Note that we index the Wikipedia pages as a whole without any distinction between the different fields of the page.
- $IDF(q_i)$ is the *inverse document frequency* for the term q_i and is calculated as $IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$ where N is the total number of documents in the index and $n(q_i)$ is the number of documents containing q_i.
- $f(q_i, W)$ is the q_i term frequency for the Wikipedia article of the candidate entity.
- k_1 and b are constants, 1.2 and 0.75, respectively.
- $|W|$ is the length of the Wikipedia article in words.
- $avgl$ is the average document length in the index containing the Wikipedia articles.

Thus, the similarity score for each candidate is computed, and every candidate is discarded except the best N. In our experimentation we found that N = 7 is a reasonable number of candidates that allows preserving coherence in the graph building step and prevents discarding nodes that are the correct solution to the mention linking problem.

The complexity of Algorithm 2 is $\mathcal{O}(n^2)$, determined by the loops in lines 3, 8, 15 and 35. The first loop has $\mathcal{O}(n)$ complexity, bounded by the number of mentions. Its nested loop has also $\mathcal{O}(n)$ complexity, in this case, bounded by the number of exact matches found in the index (even though the number of uris is often very low). The loops from lines 15 and 35 have also $\mathcal{O}(n)$ complexity. These loops are bounded by the total number of candidate uris and by the number of candidate entities found in each step of the disambiguation procedure. Note that the three nested loops, in lines 3, 8 and 15, have complexity $\mathcal{O}(n^2 + n^2 + n^2)$ which, in practice, is $\mathcal{O}(n^2)$. The disambiguation procedure in line 37 has complexity $\mathcal{O}(1)$, bounded by the constant MAX_DISAM_DEPTH.

4.3. Graph expansion

The next step in the annotation process involves interconnecting every candidate entity generated from different mentions of the text. Formally, let c_j be the candidate entity j for a mention m and C_{m_j} the set of candidate mentions for mention m_j, then, for each $(c_i, c_j) \in C_{m_i} \times C_{m_j}, m_i \neq m_j$, ABACO creates a DBpedia sub-graph that connects c_i to c_j, that is, every two nodes of different mentions are linked together if possible. The result of this linking is a unique graph in which, ideally, all candidate nodes are interconnected, although the graph may contain several isolated sub-graphs when

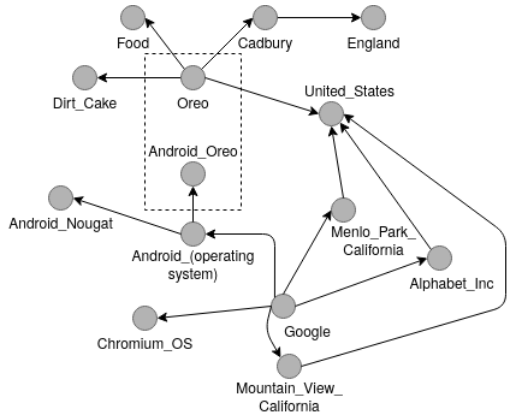

Figure 3: An example of a graph with two candidate nodes for the mention "Oreo".

topics are unrelated or even isolated nodes where there is no way to link the mention with the main graph.

Figure 3 shows a simplified example in which graphs that connect different candidate nodes are combined in a unique graph. As shown in the figure mentioned above, mentions have one or more candidate nodes, such as the entities "Oreo" and "Android Oreo" (framed in a dotted rectangle) for the mention "Oreo". These two candidate nodes exemplify the assumption on which this proposal is built: candidate nodes closely related to the document's topic are more interlinked or at least should be located closer within the graph. In this case, the candidate nodes "Android Oreo" and "Android operating system" are directly linked, while "Oreo" is at a distance of four links.

4.3.1. Explored properties

Table 1 shows some relevant properties between entities in DBpedia 2016-10. All the relations of DBpedia has been used to interconnect nodes, except four of them. Specifically `dbo:wikiPageRedirects`, which stores different URIs referring the same entity, `dbo:wikiPageDisambiguates`, which points out to other entity in conflict with the current entity because of name ambiguity, `dbo:wikiPageWikilink`, which contains other internal links of an entity in the Wikipedia page, and `rdfs:type`, which contains the type of the entity, were discarded. As aforementioned in section 4.1, `dbo:wikiPageRedirects` and `dbo:wikiPageDisambiguates` are the basis for creating the index used to identify mentions. `dbo:wikiPageWikilink` was not considered because it has no clear semantical meaning, introducing unnecessary noise to the annotation process; and finally, `rdfs:type` was not considered because `dcterms:subject` provided a more precise classification and `rdfs:type` would allow interconnecting nodes that are not really related between them just because the nodes have the same type.

Table 1: Number of instances of the most used relations included in DBpedia 2016-10.

Property	**# Instances**
`dbo:wikiPageWikilink`	48,704,874
`dcterms:subject`	13,610,094
`rdfs:type`	11,168,302
`dbo:wikiPageRedirects`	5,074,113
`skos:broader`	1,463,237
`dbo:wikiPageDisambiguates`	1,004,742
`dbo:birthPlace`	979,163
`dboprop:subdivisionType`	598,602
`dbo:country`	557,587
Other relations	20,304,358

4.3.2. Graph building

Given a source node n_s and a target node n_t, the objective is to identify the set of possible paths in the directed graph from n_s to n_t. The only parameter needed to perform the exploration is the depth of exploration, d. The higher the depth of exploration is, the denser is the sub-graph extracted from the KB, and hence higher time consumption. However, building graphs at a higher depth does not have to provide necessarily more information. These connections are often more general as the depth of exploration increases and give less information than other relations of the KB.

The algorithm implements a *bidirectional strategy* to speed up the graph building process. This strategy runs two simultaneous searches, one forward from the source node, and one back from the target node, stopping when the maximum depth of exploration, d, is reached. Thus, we retrieve every possible path between n_s and n_t that is of length d or less. Note that ABACO needs to retrieve every possible between each pair of nodes for the calculation of the centrality to be accurate. Otherwise, it would miss relations that are present in the KB but that are not considered in the calculation of the centrality.

With the aim of improving the performance of the search process, paths between every two nodes in the DBpedia have been indexed. Specifically, paths up to $i = 4$ were preprocessed. Experimentation showed that paths with more than $i = 3$ give too much indirection in the graph building step, so paths up to $i = 2$ were indexed. Table 2 gives an idea of the magnitude of this indexation. This allows that, in the best case, the graph is built in a single step when $d = 2 * i$, where i is the maximum depth indexed in the path database.

Table 2: Number of paths between entities in DBpedia 2016-10.

Depth	# Paths	Size (in bytes)	
		Raw	Processed
$i = 2$	$3.76 \cdot 10^{08}$	8.4 GB	4.3 GB
$i = 3$	$3.11 \cdot 10^{09}$	100.0 GB	54.0 GB
$i = 4$	$3.10 \cdot 10^{10}$	1.3 TB	710.0 GB

The *raw* column shows the real size of a file containing all paths for $i = 2, 3, 4$. The *processed* column uses an encoding scheme to reduce the size of URIs.

As shown in table Table 2, there are more than $3.10 \cdot 10^{10}$ paths for $i = 4$. The size (in bytes) also grows exponentially as the distance between nodes increases. To minimize this issue, we considered implementing a strategy where URIs and paths were encoded with as much as 4 bytes per URI and as less as 1 byte per URI. The improvement ratio was around 50%, although size could be further reduced by compressing URIs and paths using some of the compressors currently available (such as `xz` or `bzip2`) increasing the computation time. However, this solution was discarded since it would increase the query time *per* graph dramatically since each node to decode would imply, in the worst case, a database query. Thus, the paths are directly used without any encoding.

To store the paths as efficiently as possible, a fine-grained tuned PostgreSQL[6] database is used and an B+ indexing is performed over the source and destination nodes. A relational database is used since: *(i)* retrieving data from this database is faster than to use a database such as Cassandra[7] or MongoDB[8] that have higher latency; *(ii)* it is faster to store the data in a single simple indexed denormalized table than in a distributed schema; *(iii)* since PostgreSQL 10 parallel query[9] aids to retrieve data faster; and *(iv)* a join must be performed to retrieve the category paths of the graph and this operation is much more efficient in a relational database.

The bidirectional approach takes advantage of this indexation to reduce the exploration and retrieve in one step the full graph for a given distance d. Thus, the full list of preprocessed paths C is retrieved where $c_i \neq c_j$, that is, paths where the source and destination nodes are different. Furthermore, two different kinds of paths can be distinguished:

- *Category paths*. These paths only involve categories. The common category is searched between each pair of nodes. Thus, when indexing paths up to level $d/2$ of depth, we can build graphs with paths up to length d after performing the join between the two source columns of the indexed paths. This kind of path is the most common in the graph building since DBpedia categories may agglutinate lots of relationships (e.g, `Cat` → `(dcterms:subject)` → `Category:Animal`).
- *Direct paths*. These paths involve relationships without categories. No join is performed between direct paths since it would increase the indirection level of a relationship too much. These kind of paths are much less common in the graph and involve much more significant relationships between candidates (e.g, `Stephen_Hawking` → `(dbo:birthplace)` → `Oxford`).

Note that, first, paths are retrieved from the database and, then, the internal representation of the graph is built over an iteration of these paths. This procedure is much faster than iterating over each pair of candidates since many of them would not be connected and, thus, they would be isolated in the final graph.

4.4. Calculation of topic coherence

The concept of *degree of centrality* is used to identify the most important nodes within a graph, defined as the number of relations incoming and outgoing upon a node (directed graphs). Incoming relations are interpreted as the popularity of a node, while outgoing relations provide information about how much the node is linked to other nodes. Furthermore, paths that are too indirect or are too broad are penalized by taking into account the length of each path and the number of category nodes that are present in them.

Let $G = (V, E)$ be a graph given by a set of nodes V and relations E. The degree of centrality of a node $v \in V$ is defined as:

$$C(v) = \sum_{i=1}^{P} \frac{deg_i^-(v) + deg_i^+(v)}{(C_i + L_i)} = \sum_{i=1}^{P} \frac{1}{(C_i + L_i)} \quad (4)$$

Where P denotes the set of *paths* involving v and other candidate entities from the graph, $deg_i^-(v)$ and $deg_i^+(v)$ denote the indegree and the outdegree of the node v with respect to the path P_i, and C_i and L_i denote the number of category nodes and the length of the path P_i that involves v respectively. In practice, the numerator $deg_i^-(v) + deg_i^+(v)$ denotes the incidence of the path P_i in the node v, independently of the direction of this incidence (incoming or outgoing). Since each path can

[6] `https://www.postgresql.org/`
[7] `http://cassandra.apache.org/`
[8] `https://www.mongodb.com/`
[9] `https://www.postgresql.org/docs/10/parallel-query.html`

incide exactly once in a node, either by an outgoing relation or by an incoming relation, the numerator for each path is always 1. The denominator of the equation penalizes nodes that have many relationships with categories of indirect paths (which would be the case of popular nodes, such as countries). Note that this centrality is calculated at a *path* level instead of a *relation* level. This means that if the same relation is present in different paths, it would be counted as many times as it is present in the different paths, regardless if it is the same relation.

For example, let A and E, be candidate entities; B, C and D, be intermediate non-category nodes between A and C; and R be the set of relations $R = \{A \rightarrow E, A \rightarrow B, B \rightarrow C, C \rightarrow E, B \rightarrow D, D \rightarrow E\}$, then P would be the set of paths $P = \{A \rightarrow E, A \rightarrow B \rightarrow C \rightarrow E, A \rightarrow B \rightarrow D \rightarrow E\}$. The value of the centrality of A, $C(A)$, is calculated as follows:

$$C(A) = \frac{1}{L_{P_1}} + \frac{1}{L_{P_2}} + \frac{1}{L_{P_3}} = \frac{1}{1} + \frac{1}{3} + \frac{1}{3} \approx 1.667 \quad (5)$$

Other centrality measures, such as *PageRank* [49] and *HITS* [56], were also considered in this paper. However, they were discarded after the experimentation as they did not improve the results obtained by the degree of centrality. In addition, the algorithm complexity to compute the degree of centrality is $O(n)$, which is lower than the complexity $O(n^2)$ of *PageRank* or *HITS*.

4.5. Adjustment factor

We perform two adjustments to the relevance calculated in equation 2:

- A small fixed amount (α) is added to this relevance if the candidate matches with a candidate generated as an exact match of the mention. In this case, it is assumed that a mention can be usually linked to the most obvious entity. This value is set to 25% of the common values of textual similarity obtained with medium texts when 5-6 nodes are annotated. Note that this value is estimated empirically and, in practice, an alpha higher or shorter than this value would have no effect whatsoever on the annotation results. Furthermore, when more nodes are annotated, and the texts grow in size, the effect of α is negligible, relying entirely on the textual and centrality measures.
- Nodes that were generated by querying the Wikipedia API are discarded under the examination of two thresholds: *(i)* a textual similarity threshold and an *(ii)* relation threshold. The relation threshold is set to a low value to force that the candidates generated by this way are slightly connected with the main graph. The textual similarity threshold is set to a low value taking into account the scoring of textual similarity of medium-sized texts with, approximately, 10 mentions to annotate. These thresholds are necessary since querying Wikipedia adds much noise for adding nodes that are not directly related to the text. The combination of these thresholds avoids discarding nodes that are not related to the conjoint graph but have a high degree of textual similarity.

4.6. Combinatorial problem

Recalling from section 3.4, one of the main problems of the graph-based disambiguation approaches is the combinatorial explosion that happens when exploring the KB. To alleviate this combinatorial explosion three main strategies are performed:

(i) The number of possible candidates for each mention is kept limited by a filtering based on the context compatibility. This filtering is performed before building the disambiguation subgraph. Thus, the time to retrieve the paths between pairs of candidate entities does not depend on the number of candidate entities retrieved from the disambiguation pages.

(ii) The bidirectional strategy reduces the number of nodes that ABACO needs to explore to build the disambiguation sub-graph. In the case of a regular breadth-first search, the number of nodes needed to explore in the worst case for each pair of entities is $\mathcal{O}(b^d)$, where d is the exploration depth and b is the branching factor. In the case of a bidirectional search, the number of nodes needed to explore for each pair of entities is twice $\mathcal{O}(b^{d/2})$ which is much less than the breadth-first scenario.

(iii) The indexation of paths allows retrieving paths between candidate entities faster. Given an exploration depth, d, there are three possible scenarios, depending on the maximum indexed path length:

  – In the first scenario, when the maximum indexed path length is equal or greater than the maximum exploration depth, d, the paths between candidate entities can be retrieved in a single step by directly querying the path index ($\mathcal{O}(1)$).

– In the second scenario, when the indexed path length is equal or greater than $d/2$ and less than d, it is also possible to retrieve the paths in a single step by joining the index of paths with itself. This operation is called a "hash join" since it is a join performed over the B+ indexed columns of the path index. The complexity of this is $\mathcal{O}(N + N)$, which, in practice is $\mathcal{O}(N)$.

– In the third scenario, when the maximum indexed length is less than $d/2$, the complexity for each pair is $\mathcal{O}(b^{d/2m})$, where m denotes the maximum indexed path length.

Summarizing, in ABACO disambiguation is addressed as a holistic problem: a DBpedia sub-graph, containing the candidate nodes of all mentions, is created, and the best candidates are selected based both on their topic coherence and their context similarity. Other approaches that use graphs to disambiguate, only explore candidate nodes up to a distance of 2 or 3 relations. This is usually an issue since nodes in DBpedia are usually linked at a more profound distance, and thus graphs can be composed of isolated nodes. Our approach minimizes this issue since it accounts for all nodes up to distance 8, which, in combination with the high branching factor of DBpedia, ensures the connectivity between nodes. The combination of paths indexing and a bidirectional search strategy minimizes the cost of exploration. Finally, a new relevance model for ranking nodes has been defined that combines the degree of centrality, to measure the importance of the entity in the graph, and a similarity measure between the text-based relations of the entity and the document's mentions.

5. Evaluation

5.1. Baseline for precision and recall

To perform the evaluation of ABACO, the GERBIL [43] platform (version 1.2.7) is used with the *D2KB* (Disambiguation to KB) experiment in which the annotator shall link a specific set of mentions from a given text to their corresponding URI in a known KB or return NIL if no entry can be found in the target KB. Let M^* be the set of annotations consisting of the ground truth associated with a given set of mentions Q and let M be the set of annotations returned by an annotator. Then, the evaluation metrics used in this paper are computed as follows:

Table 3: Dataset characteristics, where *Num docs* is the number of documents in the dataset, *Entities* is the total number of entities found in the dataset, *Avg. Ent. /Doc.* is the average number of entities per document and *Avg. Wd. /Doc.* is the average number of words per document

Dataset	**Num docs**	**Entities**	**Avg.Ent./ Doc.**	**Avg.Wd./ Doc.**
ACE2004	57	253	5.37	373.9
AIDA/CoNLL Test B	231	5616	24.31	176.85
AQUAINT	50	747	14.54	220.5
IITB	103	18308	109.22	639.7
MSNBC	20	747	37.35	543.5
N3-Reuters-128	880	128	4.85	123.8
N3-RSS-500	500	1000	1.00	31.0
OKE 2015 Task 1 Ev.	101	664	6.57	30.34
OKE 2016 Task 1 Ev.	55	340	6.18	30.33
OKE 2018 Task 1 Tr.	60	378	6.3	66.37
OKE 2018 Task 2 Tr.	56	423	7.55	66.5
OKE 2018 Task 4 Tr.	100	305	3.05	23.81
Spotlight	58	330	5.69	28.6

$$P = \frac{|M \cap M^*|}{|M|}$$
$$R = \frac{|M \cap M^*|}{|M^*|}$$
$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

Where P is the precision, R is the recall, and F1 is the F1 score. Note that, in the GERBIL platform, two URI sets are matching and, thus, considered a true positive whether some of the following conditions occurs: *(i)* the URIs are classified from a known KB, and both sets match or *(ii)* both sets are classified as unknown to the KB. Furthermore, GERBIL distinguishes between macro and micro measures. Micro measures make the average of the sum of all true positives and false positives to calculate a single metric. Macro measures make the average of the precision of every document, calculated individually. In practice, micro measures show the performance of the annotator over the whole dataset, whereas macro measures show the average performance for each document. In this paper, the F1 micro score will be used as a performance metric.

5.2. Datasets

This evaluation is performed over 13 well-known datasets crafted from various sources. Datasets composed of short texts are not used since they are not the target of our annotation solution, more oriented to medium to large-sized texts from which a better context can be extracted. The main characteristics of these datasets are shown in Table 5.

- In *AQUAINT* [8] texts are collected from a news service. Only the first term occurrence of an entity is annotated, and only relevant entities are considered.
- In *MSNBC* [7] texts are collected from the news of the *MSNBC news* network. The dataset lists only the most relevant entities and where they are mentioned in the text.
- *IITB* [57] contains more than one hundred of manually annotated texts, collected from sports, entertainment, science, technology and health websites. In IITB all the mentions of an entity in the text are annotated, including entities less relevant.
- *ACE2004* [13] is composed of news, where only named entities are annotated, ignoring common names.
- *AIDA/CoNLL* [29] documents are news obtained from the *Reuters Corpus V1*, in which a subset of named entities are annotated and common names are ignored. The dataset has three subsets: *Training*, *Test A*, and *Test B*. Since the AIDA system, proposed in the same publication, has been trained using the first two and PBOH maximizes its hyperparameters using the second dataset (among other 1,000 random Wikipedia pages) only the last set was considered to carry out the experiments.
- *Spotlight* [18] contains a set of short news in which relevant entities are annotated.
- *N3-RSS-500* and *N3-Reuters-128* [58] consist, on one hand, in data scraped from 1457 RSS feeds and, on the other hand, 128 news articles sampled from the Reuters-21578 news articles dataset. Both datasets are manually annotated by domain experts. The news is gathered from all major worldwide newspapers and comprise a wide range of topics.
- *OKE* datasets were used in the OKE challenges [59, 60, 61]. These datasets contain medium-sized texts that were curated manually.

All datasets are in English.

5.3. Annotation Systems

Although many annotation systems have been proposed in the literature, only a few of them are publicly available, integrated in the GERBIL platform and currently working. In addition to ABACO, our experimentation also included the following annotation systems aforementioned in Section 3[10]: *AIDA* [29], *DBpediaSpotlight* [18], *AGDISTIS/MAG* [42], *Babelfy* [33], *FOX* [45], *FREME NER* [48], *Kea* [47], *PBOH* [44], and *OpenTapioca* [41].

Unfortunately, some of the annotators are not currently working with the online version of GERBIL even though they are operating normally [11]. Therefore, a local version of GERBIL 1.2.5 has been deployed to solve those issues. The the results of *TagMe2* and *NERD-ML* correspond with the local deploying of GERBIL (marked as "*" in the result table).

6. Results

In this section the results of the tests of ABACO and other eleven entity annotation systems described in the previous section are detailed. These systems have been tested over the datasets listed in Table 3 without any tuning or training. Furthermore, the performance of ABACO with different graph weighting measures, HITS [56] and PageRank [49], is also analyzed.

In the following subsections, the effectiveness of these systems is presented, in terms of F1, and discuss the performance of ABACO for the D2KB problem.

6.1. Results and discussion

The comparison for the D2KB problem evaluated in GERBIL is listed in Table 6.1.[12]The comparison for the D2KB problem evaluated in GERBIL is shown in Table 6.1, where ABACO obtains the best results in 11 of 13 datasets and the best mean score of all annotators that have been evaluated.

The next best annotators are PBOH and Kea. As far as the differences with these annotators is concerned, there are some datasets where PBOH performance is heavily diminished when it is compared to ABACO in terms of the F1 score, such as N3-RSS-500 (third-best annotator with 14.12% difference), OKE 2015 dataset (second-best annotator with 8.89 % difference), OKE

[10]Baseline shown in this paper is available in the following link `http://gerbil.aksw.org/gerbil/experiment?id=201903070014`. Baseline for the OpenTapioca annotator is available in the following link `http://gerbil.aksw.org/gerbil/experiment?id=201909120008` (this annotator was added after the main baseline had been performed).

[11]See issue `https://github.com/dice-group/gerbil/issues/312`

[12]Results are available in the following link: `http://gerbil.aksw.org/gerbil/experiment?id=201903010008`

	ACE2004	IITB	AQUAINT	Spotlight	MSNBC	N3-Reuters-128	N3-RSS-500	OKE 2015 Task 1 evaluation set	OKE 2016 Task 1 evaluation set	AIDA/CoNLL-Test B	OKE 2018 Task 1 training dataset	OKE 2018 Task 2 training dataset	OKE 2018 Task 4 training dataset	**Mean F1**
AGDISTIS/MAG	66.34	30.63	42.50	20.74	73.19	64.66	61.50	56.02	49.12	47.17	70.37	62.41	82.62	55.94
AIDA	70.29	17.74	54.83	24.51	68.33	46.30	45.21	55.56	49.47	68.31	69.03	59.10	73.00	53.97
Babelfy	55.78	36.53	67.79	51.59	70.34	44.85	44.34	57.98	54.33	67.60	62.46	55.72	67.66	56.69
DBpedia Spotlight	40.66	29.26	49.77	67.82	38.38	28.94	18.72	30.78	38.96	48.74	36.26	35.86	42.96	39.00
FOX	65.12	11.46	37.98	14.36	59.62	58.62	59.21	55.11	46.65	53.28	65.01	55.15	75.51	50.54
FREME NER	5.000	24.89	53.49	52.46	43.22	24.62	27.77	31.40	36.18	34.45	30.67	36.90	48.47	38.04
OpenTapioca	56.95	9.080	44.26	17.53	48.60	37.89	31.63	46.94	48.26	43.29	52.65	46.23	62.80	42.01
Kea	70.67	40.26	78.29	71.90	72.70	52.54	45.11	61.92	63.59	59.52	71.63	66.41	80.76	64.25
NERD-ML*	57.37	42.60	59.59	55.16	60.73	40.95	37.39	61.44	63.14	0.320	0.000	0.000	8.180	37.45
TagMe 2*	71.71	36.26	71.85	65.76	63.52	43.84	47.07	57.56	63.48	57.67	59.41	53.62	71.12	58.68
PBOH	77.30	36.46	**81.29**	78.47	81.53	64.77	53.45	62.80	68.53	**74.84**	69.44	64.11	81.97	68.84
ABACO	**78.10**	**44.59**	76.89	**83.03**	**85.14**	**69.77**	**67.57**	**71.69**	**73.24**	70.46	**76.72**	**70.21**	**84.59**	**73.23**

Table 4: Micro F1 measures for 13 datasets evalutated against 11 annotation systems and mean F1 score for each annotator. We highlight in **bold** the best system in each dataset. Systems marked as "*" were tested on a local deployment of GERBIL 1.2.5

	ACE2004	IITB	AQUAINT	Spotlight	MSNBC	N3-Reuters-128	N3-RSS-500	OKE 2015	OKE 2016	AIDA Test B	OKE 2018 Task 1	OKE 2018 Task 2	OKE 2018 Task 4	**Mean F1**
ABACO	78.1	44.59	76.89	83.03	85.14	69.77	67.57	71.69	73.24	70.46	76.72	70.21	84.59	73.23
ABACO + PageRank	76.8	39.89	77.03	83.33	84.07	70.27	67.47	69.88	73.53	67.44	72.58	67.93	83.93	71.81
ABACO + HITS	72.88	37.41	74.83	82.42	81.53	69.89	67.67	68.98	69.12	65.85	73.28	67.14	83.28	70.32

Table 5: F1 scores for 13 datasets using different graph weighting strategies in ABACO.

2018 Task 1 dataset (fourth-best annotator with 7.28% difference) or the IITB dataset (fourth best annotator with 8.13% difference). The differences are even stronger if we compare ABACO with the Kea annotator, for instance, the 12.44% difference in MSNBC dataset or the 22.46% difference in N3-RSS-500 dataset.

It is worth noting ABACO results in the *IITB* dataset, which, according to Table 6.1, is the dataset with the higher number of nodes (or entities) per document, followed closely by NERD-ML (1.99% F1 score) and followed by KEA (4.33% F1 score). This result proves the robustness of the graph building algorithm when a high number of nodes is given, even when common nouns have to be annotated.

On the other hand, it is also worth noting the case of the *ACE2004* dataset, where, although ABACO obtains the best result, is closely followed by *PBOH* (1.2% F1 score) and far from the third-best followed by TagMe 2 (6.39 % F1 score). As aforementioned, the news agency in the headers of the text is identified as a mention to annotate. However, it is usually annotated incorrectly by ABACO due to the lack of contextual information in the text. Although this problem can be easily solved by implementing a parser to recognize headers, this solution was not implemented in ABACO since it would be very dataset-specific. Even with that handicap, ABACO manages to get the best score.

As an additional result of our analysis, we have identified two issues that reduce the effectiveness of ABACO. First, when the text is short sized and some of the terms are not directly correlated with the main topic of the document, ABACO is not able to link the mention to its corresponding entity due to the mismatch between topic and entity presence. Furthermore, ABACO does not always find the correct entity when there exist two or more candidate nodes that are semantically close such as `New York` and `New York City` (state and city) or `Chicago White Sox` and `Chicago Cubs` (both are baseball teams from Chicago). On the other hand, in some cases, ABACO struggles to retrieve the entity linked to some surnames when no proper candidate is found neither in the disambiguation pages, nor in the coreference resolution phase (when the surname is not mentioned explicitly with its full name). These is-

sues prevent ABACO from getting a better result in the *AQUAINT* and *AIDA* datasets.

Table 5 shows the performance of different graph weighting strategies of ABACO. Centrality weighting is changed for PageRank or HITS, but BM25 is still used for calculating the textual similarity between nodes. As shown in the aforementioned table, HITS underperforms the centrality weighting measure in every dataset except *N3-RSS-500* and *N3-Reuters-128* (by a low margin). PageRank shows more promising results by improving the results in datasets *AQUAINT*, *SpotLight*, *N3-Reuters-128* and *OKE 2016*. However, these differences are negligible in front of the performance loses in other datasets (2-3% F1 score). These differences are rooted in the internal structure of DBpedia, in which a few nodes agglutinate lots of incoming and outgoing links with category paths. These links make PageRank and HITS think that these nodes are more important than they really are, unlike our centrality measure that penalizes paths with long lengths and containing categories.

7. Conclusions

In this paper, we presented ABACO, which, contrary to most approaches, solves name ambiguity by extracting from the KB the sub-graph that best describes the document. This graph is composed of interrelated nodes, trying to maintain the coherence of the topics of the document, and thus nodes are collectively disambiguated.

ABACO has been developed to deal efficiently with massive databases and disambiguation graphs. Specifically, a bidirectional algorithm explores the KB looking for relations between candidate nodes. The exploration cost of this algorithm is drastically reduced since paths between nodes are indexed. Thus, ABACO can explore more exhaustively the KB, allowing building a knowledge-subgraph to a depth greater than other approaches. In addition, a relevance model has been defined to score nodes according to their importance in the graph (degree of centrality) as well as their relatedness with the mentions of the document (semantic similarity). Only the best candidate entity of each mention is maintained in the graph. Unlike other approaches from the state of the art, ABACO successfully builds graphs from coherent candidates (given its textual similarity) and consistently ranks nodes taking into account the inner structure of the DBpedia graphs as well as the textual similarity between texts and nodes. Furthermore, this algorithm requires no training nor tuning whatsoever.

Finally, a version of ABACO has been compared with 11 other annotators and 13 different datasets. Results show that ABACO obtains the best results in terms of F1 score in 11 of the 13 datasets.

For future work we want to research further enhancements of the centrality measure used in the paper. More specifically, we believe that taking into account the global popularity of entities in the KB or the directionality of paths in the calculation of the centrality could give a more precise measurement of the entity topic coherence. Furthermore, more distinctions between types of nodes than just "category" or "non-category" nodes could be made in the calculation of this centrality.

Acknowledgments

This work was supported by the Spanish Ministry of Economy and Competitiveness under the project TIN2015-73566-JIN and TIN2017-84796-C21-R, and by the European Regional Development Fund (ERDF/FEDER), the Consellería de Cultura, Educación e Ordenación Universitaria (accreditation 2016-2019, ED431G/08, reference competitive group 2018-2020, ED431C 2018/29).

References

References

[1] J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, R. R. Swick, Annotea: an open RDF infrastructure for shared web annotations, Computer Networks 39 (5) (2002) 589–608.

[2] A. Kiryakov, B. Popov, I. Terziev, D. Manov, D. Ognyanoff, Semantic annotation, indexing, and retrieval, Web Semantics: Science, Services and Agents on the World Wide Web 2 (1) (2004) 49–79.

[3] W. Shen, J. Wang, J. Han, Entity linking with a knowledge base: Issues, techniques, and solutions, IEEE Transactions on Knowledge and Data Engineering 27 (2) (2015) 443–460.

[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, van Patrick Kleef, S. Auer, et al., DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia, Semantic Web 6 (2) (2015) 167–195.

[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, 2008, pp. 1247–1250.

[6] R. C. Bunescu, M. Pasca, Using encyclopedic knowledge for named entity disambiguation, EACL 6 (2006) 9–16.

[7] S. Cucerzan, Large-scale named entity disambiguation based on Wikipedia data, EMNLP-CoNLL 7 (2007) 708–716.

[8] D. Milne, I. H. Witten, Learning to link with Wikipedia, Proceedings of the 17th ACM conference on Information and Knowledge Management (2008) 509–518.

[9] X. Han, J. Zhao, Named entity disambiguation by leveraging Wikipedia semantic knowledge, Proceedings of the 18th ACM conference on Information and Knowledge Management (2009) 215–224.

[10] A. L. Gentile, Z. Zhang, L. Xia, J. Iria, Graph-based semantic relatedness for named entity disambiguation, in: Proceedings of International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES, S3T, 2009, p. 13.

[11] P. Ferragina, U. Scaiella, Tagme: on-the-fly annotation of short text fragments (by Wikipedia entities), Proceedings of the 19th ACM international conference on Information and knowledge management (2010) 1625–1628.

[12] B. Hachey, W. Radford, J. R. Curran, Graph-based named entity linking with Wikipedia, in: Web Information System Engineering–WISE 2011, Springer, 2011, pp. 213–226.

[13] L. Ratinov, D. Roth, D. Downey, M. Anderson, Local and global algorithms for disambiguation to Wikipedia, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Association for Computational Linguistics, 2011, pp. 1375–1384.

[14] X. Han, L. Sun, J. Zhao, 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, 2011, pp. 765–774.

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

[16] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann, T. Stegemann, Relfinder: Revealing relationships in RDF knowledge bases, in: Semantic Multimedia, Springer, 2009, pp. 182–187.

[17] R. Mirizzi, A. Ragone, T. D. Noia, E. D. Sciascio, Semantic tag cloud generation via DBpedia, in: E-Commerce and Web Technologies, Springer, 2010, pp. 36–48.

[18] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, DBpedia spotlight: shedding light on the web of documents, in: Proceedings of the 7th International Conference on Semantic Systems (I- SEMANTICS 2011), ACM, 2011, pp. 1–8.

[19] O. Muñoz-García, A. García-Silva, Ó. Corcho, M. H. Hernández, C. Navarro, Identifying topics in social media posts using DBpedia, in: Proceedings of the Networked and Electronic Media Summit (NEM summit 2011), 2011, pp. 1–6.

[20] S. Hakimov, S. A. Oto, E. Dogdu, Named entity recognition and disambiguation using linked data and graph-based centrality scoring, in: Proceedings of the 4th International Workshop on Semantic Web Information Management, ACM, 2012, p. 4.

[21] I. Hulpus, C. Hayes, M. Karnstedt, D. Greene, An eigenvalue-based measure for word-sense disambiguation, in: The Florida Artificial Intelligence Research Society, FLAIRS Conference, 2012, pp. 226–231.

[22] I. Hulpuş, C. Hayes, M. Karnstedt, D. Greene, M. Jozwowicz, Kanopy: Analysing the semantic network around document topics, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 677–680.

[23] D. S. Carvalho, A. Danilo, J. C. da Silva, Graphia: Extracting contextual relation graphs from text, in: The Semantic Web: ESWC 2013 Satellite Events, Springer, 2013, pp. 236–241.

[24] B. Fetahu, S. Dietze, B. P. Nunes, M. A. Casanova, D. Taibi, W. Nejdl, A scalable approach for efficiently generating structured dataset topic profiles, in: Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 519–534.

[25] A. Varga, A. E. C. Basave, M. Rowe, F. Ciravegna, Y. He, Linked knowledge sources for topic classification of microposts: A semantic graph-based approach, Web Semantics: Science, Services and Agents on the World Wide Web 26 (2014) 36–57.

[26] M. Schuhmacher, S. P. Ponzetto, Knowledge-based graph document modeling, in: Proceedings of the 7th ACM international conference on Web search and data mining, ACM, 2014, pp. 543–552.

[27] C. Fellbaum, WordNet, Wiley Online Library, 1998.

[28] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: A core of semantic knowledge, in: Proceedings of the 16th International Conference on World Wide Web, WWW '07, ACM, 2007, pp. 697–706.

[29] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, G. Weikum, Robust disambiguation of named entities in text, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2011, pp. 782–792.

[30] R. Navigli, S. P. Ponzetto, BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, Artificial Intelligence 193 (2012) 217–250.

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

[32] M. Dojchinovski, T. Kliegr, Entityclassifier. eu: Real-time classification of entities in text with Wikipedia, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 654–658.

[33] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: A unified approach, Transactions of the Association for Computational Linguistics 2 (2014) 231–244.

[34] D. B. Nguyen, J. Hoffart, M. Theobald, G. Weikum, AIDA-light: High-throughput named-entity disambiguation, in: Linked Data on the Web (LDOW2014), 2014, pp. 1–10.

[35] R. Usbeck, A.-C. N. Ngomo, M. Röder, D. Gerber, S. A. Coelho, S. Auer, A. Both, AGDISTIS-graph-based disambiguation of named entities using linked data, in: International Semantic Web Conference–ISWC 2014, Springer, 2014, pp. 457–471.

[36] I. Yamada, H. Shindo, H. Takeda, Y. Takefuji, Joint learning of the embedding of words and entities for named entity disambiguation, in: Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, Association for Computational Linguistics, 2016.

[37] O.-E. Ganea, T. Hofmann, Deep joint entity disambiguation with local neural attention, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2017.

[38] P. Radhakrishnan, P. Talukdar, V. Varma, ELDEN: Improved entity linking using densified knowledge graphs, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, 2018.

[39] P. Le, I. Titov, Improving entity linking by modeling latent relations between mentions, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Association for Computational Linguistics, 2018, p. 1595–1604.

[40] J. Raiman, O. Raiman, Deeptype: Multilingual entity linking by neural type system evolution, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 5406–5413.

[41] A. Delpeuch, OpenTapioca: Lightweight Entity Linking for Wikidata, arXiv e-prints (2019) 1–6.

[42] D. Moussallem, R. Usbeck, M. Röeder, A.-C. N. Ngomo, MAG: A Multilingual, Knowledge-base Agnostic and Deterministic Entity Linking Approach, in: Proceedings of the Knowledge Capture Conference on - K-CAP 2017, ACM Press, 2017.

[43] M. Röder, R. Usbeck, A.-C. N. Ngomo, Gerbil – benchmarking named entity recognition and linking consistently, Semantic Web 9 (5) (2018) 605–625.
[44] O.-E. Ganea, M. Ganea, A. Lucchi, C. Eickhoff, T. Hofmann, Probabilistic bag-of-hyperlinks model for entity linking, in: Proceedings of the 25th International Conference on World Wide Web - WWW '16, ACM Press, 2016.
[45] R. Speck, A.-C. N. Ngomo, Ensemble learning for named entity recognition, in: The Semantic Web – ISWC 2014, Springer International Publishing, 2014, pp. 519–534.
[46] M. V. Erp, G. Rizzo, R. Troncy, Learning with the web: Spotting named entities on the intersection of nerd and machine learning, in: # MSM, 2013, pp. 27–30.
[47] N. Steinmetz, H. Sack, Semantic multimedia information retrieval based on contextual descriptions, in: The Semantic Web: Semantics and Big Data, Springer Berlin Heidelberg, 2013, pp. 382–396.
[48] FREME NER.
URL :http://www.freme-project.eu/
[49] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web, in: Proceedings of the 7th International World Wide Web Conference, Elsevier, 1998, pp. 161–172.
[50] A. Cetoli, M. Akbari, S. Bragaglia, A. D. O'Harney, M. Sloan, Named Entity Disambiguation using Deep Learning on Graphs, arXiv e-prints (2018) 1–8.
[51] S. Khalife, M. Vazirgiannis, Scalable graph-based method for individual named entity identification, in: Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13), Association for Computational Linguistics, 2019, pp. 17–25.
[52] A. Mandalios, K. Tzamaloukas, A. Chortaras, G. B. Stamou, Geek: Incremental graph-based entity disambiguation, in: LDOW@WWW, 2018, pp. 51–60.
[53] V. C. Tran, N. T. Nguyen, H. Fujita, D. T. Hoang, D. Hwang, A combination of active learning and self-learning for named entity recognition on twitter using conditional random fields, Knowledge-Based Systems 132 (2017) 179–187.
[54] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, D. McClosky, The stanford CoreNLP natural language processing toolkit, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, 2014.
[55] K. Clark, C. D. Manning, Entity-centric coreference resolution with model stacking, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, 2015, pp. 1405–1415.
[56] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the ACM (JACM) 46 (5) (1999) 604–632.
[57] S. Kulkarni, A. Singh, G. Ramakrishnan, S. Chakrabarti, Collective annotation of Wikipedia entities in web text, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 457–466.
[58] M. Röder, R. Usbeck, S. Hellmann, D. Gerber, A. Both, N3 - a collection of datasets for named entity recognition and disambiguation in the nlp interchange format, in: LREC, 2014.
[59] A. G. Nuzzolese, A. L. Gentile, V. Presutti, A. Gangemi, D. Garigliotti, R. Navigli, Open knowledge extraction challenge, in: Semantic Web Evaluation Challenges, Springer International Publishing, 2015, pp. 3–15.
[60] A. G. Nuzzolese, A. L. Gentile, V. Presutti, A. Gangemi, R. Meusel, H. Paulheim, The second open knowledge extraction challenge, in: Semantic Web Challenges, Springer International Publishing, 2016, pp. 3–16.
[61] R. Speck, M. Röder, F. Conrads, H. Rebba, C. C. Romiyo, G. Salakki, R. Suryawanshi, D. Ahmed, N. Srivastava, M. Mahajan, A.-C. N. Ngomo, Open knowledge extraction challenge 2018, in: Semantic Web Challenges, Springer International Publishing, 2018, pp. 39–51.