Supervised Learning for Linking Named Entities to Knowledge Base Entries

Ivo Anastácio
INESC-ID Lisboa

Bruno Martins
INESC-ID Lisboa

Pável Calado
INESC-ID Lisboa

{ivo.anastacio, bruno.g.martins, pavel.calado}@ist.utl.pt

Abstract

This paper addresses the challenging information extraction problem of linking named entities in text to entries in a knowledge base. Our approach uses supervised learning to (a) rank candidate knowledge base entries for each named entity, (b) classify the top-ranked entry as the correct disambiguation or not, and (c) group together the named entities without a corresponding entry in the knowledge base. We analyze the fundamental design challenges involved in the development of a learning-based entity-linking system, and we provide extensive experimental results for a wide range of methods and feature sets. Our experiments over the datasets from the Text Analysis Conference (TAC) Entity Linking Task demonstrate the effectiveness of supervised learning methods, showing that out-of-the-box algorithms and relatively simple to compute features can obtain very competitive results.

1 Introduction

Given the large amounts of textual data currently available on the Web, research on information extraction methods to automatically extract structured information from these sources is getting increasingly popular. Information Extraction (IE) can be further divided into several sub-problems, most notably *named entity recognition* (Whitelaw et al., 2008), *relationship extraction* (Etzioni et al., 2008), and *named entity disambiguation* (Ji and Grishman, 2011). This work addresses the later sub-problem, also known as grounding, cross-document co-reference resolution, or named entity linking. It can be briefly summarized as the task of mapping an entity previously recognized by a Named Entity Recognition (NER) system, i.e. the reference, to an identifier specific to the concept that the named entity is referring to in the text, i.e. the referent. The named entity disambiguation problem is currently receiving substantial attention in information extraction community, given its recent inclusion as a specific task in the NIST-sponsored Automated Content Extraction (ACE) evaluations (i.e., the ACE-2008 cross-document co-reference resolution task) or the Text Analysis Conference (i.e. the Knowledge Base Population task, referred to as TAC-KBP). For instance, consider the following sentences, each belonging to a different textual document, and consider the word *Bush* as being recognized as a named entity:

1. ***Bush*** *won the 2004 presidential election.*
2. *President* ***Bush*** *was born in 1946.*
3. ***Bush****'s new album is expected to become a hit.*

By analyzing the context surrounding each reference to *Bush*, a named entity disambiguation system should assign the same identifier to the first two references, as they both refer to former U.S. President George W. Bush, while the named entity in the third document is referring to a popular rock band, thus corresponding to a different identifier. Although this example considered entities sharing the same name, entities that are misspelled or that can be referenced by multiple equivalent names (e.g., *New York City, NYC* and *Big Apple*) should also be assigned to the same identifier.

Possible applications for named entity disambiguation include (a) enriching documents with links to authoritative Web pages on the referenced entity, (b) grouping Web search results for queries corresponding to ambiguous entities, based on the possible referents, and (c) supporting advanced information retrieval applications such as question answering and entity search.

This work presents a thorough study on the subject of named entity disambiguation, providing a discussion of its key issues, as well as an empirical analysis of the effectiveness of different machine learning models and different sets of features. Machine learning methods for addressing ranking tasks are usually known as Learning to

Rank (Liu, 2009) approaches, and they have been successfully applied in document retrieval systems. However, their application in named entity disambiguation has received less attention. We specifically try to answer the following research questions:

- Which features lead to a more effective learning-based entity linking approach?
- When modeling the task as a learning to rank problem, what are the specificities of named entity disambiguation when compared to traditional document retrieval?
- Is it useful to train specific models according to the estimated query type (e.g., use different models for people, organizations, or location entities)?

The rest of this paper is organized as follows: Section 2 describes some of the key issues involved in the task, also presenting the usual architecture of entity linking systems. Section 3 presents related work, covering both the areas of named entity disambiguation and learning to rank. Section 4 presents the details of our machine learning approach, with an emphasis on the features we used to model the relationship between named entities and candidate disambiguations. Section 5 presents the experimental results of a study comparing different configurations for the proposed machine learning approach. Finally, Section 6 presents the main conclusions and points directions for future work.

2 Named Entity Disambiguation

Previous works on named entity disambiguation have distinguished between two types of approaches, namely *corpus-based* (i.e., with no a priori knowledge on the entities, using clustering to disambiguate the references) and *knowledge-based* (i.e., using a knowledge base with information about each entity, usually Wikipedia, and assigning references to the most similar knowledge base entry). However, competitions designed to evaluate named disambiguation systems (e.g., TAC-KBP (Ji and Grishman, 2011)) are now requiring that references should not only be assigned to a predefined knowledge base entry, but also for references to entities that are not in the knowledge base to be grouped together if they refer to the same concept, thus mixing these two types of approaches.

Although the knowledge-based approach can be seen as a ranking problem, where the named entity reference is the equivalent to a query and the assigned referent should be the highest ranked candidate, the named entity disambiguation task has several specificities, namely:

- **Binary relevance and single (or no) answer**. Contrary to document retrieval where documents can be more or less relevant to a query, here only one knowledge base entry can be the correct referent to a given reference, or none if the entity is not represented in the knowledge base.
- **Imbalancement problems**. In document retrieval applications we usually have, for each query, several examples of relevant documents and some examples of irrelevant documents, which are used to train a ranking model. However, in the named entity disambiguation task, we have at most one relevant knowledge base entry and many irrelevant entries.
- **Optimization of evaluation metric**. While recent learning to rank models for document retrieval directly optimize an evaluation metric such as MAP, MRR or NDCG, for named entity disambiguation systems following a learning to rank approach, the P@1 seems a more suitable evaluation metric. As an illustration consider two models, namely one that ranks the correct referent first for one query and last for all others, and another which always ranks the correct referent second. Although the second model places the correct referents very close to the top, it would still perform worse than the model that assigns a correct disambiguation only once.

The general architecture for named entity linking systems, with basis on an analysis of several approaches found in related literature, is presented in Figure 1. This architecture consists of five main modules:

1. **Query expansion**: Knowledge base referents might be referenced in the texts by several alternative names, some of which might be more ambiguous than others. Therefore, given a reference, most systems apply expansion techniques that try to identify other names used in the source document that reference the same entity.
2. **Candidate generation**: This module filters the knowledge base entries that might correspond to the query, based on string similarity. Since Wikipedia is the most commonly used knowledge base, some of its hyperlink structure is also widely used to obtain alternative names (e.g., disambiguation pages, redirect pages, anchors).
3. **Candidate ranking**: This module sorts the retrieved candidates according to the likelihood of being the correct referent. Most approaches use either a *Learning to Rank* approach (Liu, 2009) or an heuristic method such as the *Vector Space IR Model* (Baeza-Yates and Ribeiro-Neto, 1999).
4. **Candidate validation**: This module decides whether the top ranked referent is an error resulting from the fact that the correct referent is not

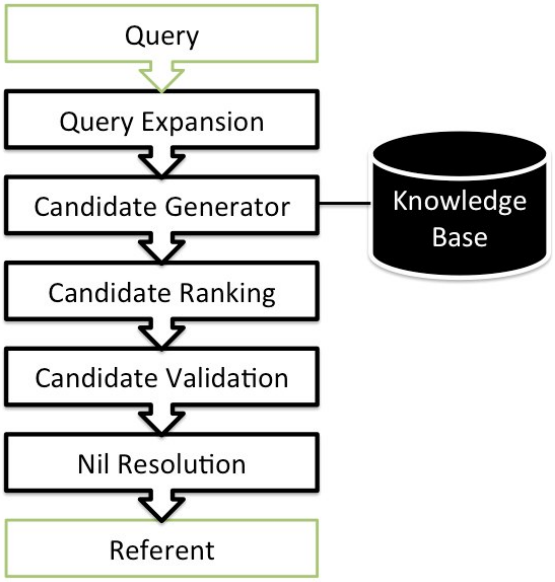

Figure 1: The modules that compose the general architecture of a complete named entity disambiguation system.

present in the knowledge base. Commonly used approaches include setting a threshold on the ranking score, training a specific classification model, or applying a voting scheme.

5. **Nil entity resolution** If the system decides that a given query has no corresponding match in the knowledge base (aka, a *nil query*), then, the nil resolution module should output an identifier specifically generated for all references to that particular entity. Nil resolution is a less studied problem, with little or no previous discussions about it.

3 Related Work

This section presents the most important related works concerning the subjects of (a) named entity disambiguation, and (b) supervised learning to rank.

3.1 Named Entity Disambiguation

The seminal work by Bagga and Baldwin (1998) addressed the named entity disambiguation task through the clustering of all references corresponding to the same entity, using the Vector Space Model (Baeza-Yates and Ribeiro-Neto, 1999). Although the authors only tried to resolve person names, their approach also seems suitable for other entity types. The specific methodology of this approach first looks for the co-reference chains of each document, i.e. the lists of all mentions to a given entity made in each single document. Then, for each co-reference chain, the system produces an entity summary consisting of all noun phrases where a chain element occurs. Finally, the similarity between all cross-document summaries is measured and summaries with a cosine similarity score above a given threshold are considered to be about the same entity.

However, an increasingly popular trend in the area of named entity disambiguation relates to assigning each entity mention to the corresponding Wikipedia entry, rather than to a cluster with equivalent entities. This new representation makes it possible to model the named entity disambiguation task as a ranking problem, where each mention is assigned to its most similar Wikipedia page. The work by Bunescu and Bunescu and Pasca (2006) is one of the earliest and most notorious proposals following this methodology. The authors developed a linear similarity function which considered contextual and categorical features, with weights optimized using supervised learning. The authors also addressed the problem of finding the correct referent for entity mentions not included in Wikipedia (i.e., the nil entities). The proposed solution involved defining a similarity threshold below which no assignment was performed by the system.

The work by Cucerzan (2007) presents a named entity disambiguation approach which also relies on Wikipedia as an external knowledge repository. Contrary to most other approaches, Cucerzan considers, as context for each entity reference, not the surrounding words, but instead the remaining entity references made in the same document. The proposed approach uses the traditional Vector Space IR Model, comparing document vectors with the referent vectors. The document vector contains the categories of all possible referents for all entity references found in its text, as well as the number of occurrences of each reference. The referents have binary feature vectors with all the categories and entity references found in its Wikipedia entry. Interestingly, the similarity measure used by the author does not normalize the feature values, thus privileging important entities, which tend to have longer descriptions, more mentions, and more categories. Also, the author argues that the errors originating from the usage of the *one sense per discourse* principle (Gale et al., 1992), which simplifies the disambiguation problem through the assumption that a given document does not contain homonym entities, are non-negligible. His approach to address this problem involved determining a reference's context in an iterative fashion — whenever more than one Wikipedia entity scored higher than a predefined threshold, the considered context would be shrunk to the level of a paragraph, and possibly to the level of a sentence.

Previous works have also addressed disambiguation tasks focusing on specific types of entities. For instance Leidner surveyed a variety of approaches for place reference resolution (Leidner, 2007). He concluded that the

most effective methods usually rely on gazetteer matching for performing the identification (e.g., matching text expressions against the Geonames database), together with natural language processing heuristics such as default senses (i.e., disambiguation should be made to the most important referent, estimated with basis on population counts) or geographic heuristics such as the spatial minimality (i.e., disambiguation should minimize the bounding polygon that contains all candidate referents).

Mihalcea and Csomai (2007) proposed the Wikify! system for performing word sense disambiguation based on Wikipedia articles. Their approach involves four main steps, namely one for selecting the candidate referents, two disambiguation modules that independently determine the most probable referent, and a fourth step that checks if the disambiguation modules agree. If there is no agreement for the most probable reference, then no referent is assigned (i.e., we have a nil entry). On what regards the disambiguation modules, we have that one of them measures the contextual overlap between the reference and candidate referents, while the other leverages on the manually assigned links inside Wikipedia articles to train a supervised learning approach based on a Naïve Bayes model. The feature vectors include not only the word terms, but also their parts of speech.

Sarmento et al. studied the development of a Web-scale named entity disambiguation system, discussing several scalability issues (Sarmento et al., 2009). The authors argue that not only is there a large number of entity references, as their distribution is highly skewed, since for each named entity there is usually a small set with of popular real-world concepts, to which the vast majority of entity mentions refers to, and there are many entities with much less information and more noise. Given the aforementioned characteristics, Sarmento et al. suggested the usage of a clustering algorithm capable of handling unbalanced data distributions, and with a stopping criteria that does not depend on a predefined value for the number of clusters. The authors ended up following a graph-based clustering approach, where entity mentions (i.e., nodes) are linked together if their similarity is above a given threshold. The clusters are then directly obtained by looking at the connected components. Their performance results showed that name co-occurrence information was not sufficient for merging distinct facets of the same entity. This became obvious after manually inspecting the results for dominant entities, which had multiple clusters, each representing a facet of an entity. They also experimented with increasingly larger samples of data and noticed that, as the size of the dataset got bigger, the complexity of the disambiguation task increased, since the number of entities and/or their respective scopes was also larger.

Following the success of learning to rank approaches in document retrieval, Zheng et al. evaluated learning to rank methods in the named entity disambiguation task (Zheng et al., 2010). The considered ranking methods included representative algorithms of pointwise (i.e., SVM regression and Perceptron), pairwise (i.e., Ranking Perceptron), and listwise (i.e., ListNet) learning to rank approaches — see Section 3.2. The authors used approximately twenty features to represent candidates, divided into three groups, namely (a) surface features, which measure the name similarity between the reference and the candidate referents, (b) context features, that measure the context similarities, and (c) special features, which represent an entity's geographical and categorical aspects. Experiments showed that the listwise approach was the most successful. In order to address the cases where references had no correct referent in the knowledge base, the authors supply the top ranked referent to a binary classifier which, using a set of features very similar to the ones used for ranking, decides whether that referent is correct or not.

He and de Rijke (2010) compared the effectiveness of learning to rank methods (i.e., AdaRank and Ranking SVM) against traditional classification approaches (i.e., SVM, J48, and Naïve Bayes) in the task of suggesting relevant Wikipedia articles to a given concept or entity found in a text. Their results support the advantages of learning to rank over traditional classification. AdaRank obtained the best results in all evaluation measures, including MAP, P@1, and P@5.

3.2 Supervised Learning to Rank

Previous works in information retrieval have addressed the usage of supervised machine learning for developing search engine ranking formulas, combining multiple estimators for document relevance in an optimal way. Both Liu (2009) and Li (2011) presented good surveys on the subject of learning to rank (L2R) for information retrieval, categorizing the existing algorithms into three groups, according to their input data representation and their optimization objectives:

- **Pointwise approach** - The L2R task is seen as a regression or a classification problem. Given feature vectors of each single resource from the data for the input space, the relevance degree of each individual resource, towards an input query, is predicted with scoring functions based on classification or regression models. Through these scores, we can sort resources and produce the final ranked list. Several pointwise methods have been proposed, including Multiclass Classification for Ranking (McRank) (Li et al., 2007) or Initialized Gradient Boosted Regression Trees (IGBRT) (Mohan et al., 2011).

- **Pairwise approach** - The L2R task is seen as a binary classification problem for pairs of resources, since the relevance degree can be regarded as a binary value telling which ordering of the resources is better for a given pair. Given feature vectors of pairs of resources from the data for the input space, the relevance degree of each of those resources can be predicted with scoring functions which try to minimize the average number of misclassified pairs. Several different pairwise methods have been proposed in the related literature, such as SVM$rank$ (Joachims, 2002) or Ranking Perceptron (Collins and Duffy, 2002).

- **Listwise approach** - The L2R task is addressed in a way that takes into account an entire set of resources associated with a query. These methods train a ranking function through the minimization of a listwise loss function defined on the predicted list and some ground truth list. Given feature vectors for a list of resources from the input space data, the relevance degree of each of those resources can be predicted with scoring functions which try to directly optimize the value of a particular information retrieval evaluation metric computed over the predicted list, averaged over all queries in the training data (Liu, 2009). Several different listwise methods have also been proposed, including ListNet (Cao et al., 2007) and AdaRank (Xu and Li, 2007).

In this paper, we experimented with several pairwise and listwise state-of-the-art learning to rank algorithms available through open-source tools.

In the past, and noticing that significant differences may exist between queries, several authors have proposed to use multiple ranking models, specific to the type of query, in order to improve L2R effectiveness. For instance Geng et al. (2008) proposed a query-dependent L2R method which dynamically creates a ranking model for a given query by using the k nearest training queries and their corresponding query-document feature vectors, afterwards using this model to rank the documents with respect to the query. Their experimental results showed that query-dependent ranking outperformed the baseline method of using a single ranking function, effectively leveraging the useful information of the similar queries and avoiding the negative effects from the dissimilar ones. Zhu et al. (2009) demonstrated that it is highly beneficial to divide queries into multiple groups and address ranking problems through multiple models based on query difficulty. Experiments showed that by using a classification model to predict query difficulty, latter using Ranking SVM or RankNet to build specific L2R models for each difficulty class, one can achieve significant improvements in the task of web search ranking. Similar ideas are explored in this paper, by also experimenting with entity-specific ranking models to select the best candidate disambiguation (i.e., one ranking model for entities of the type person, one for organizations, and another for geo-political entities).

4 Using Supervised Learning for Named Entity Disambiguation

In order to study the main aspects influencing the performance of entity linking systems, we developed a fully functional prototype that includes all the five modules from the general architecture shown in Figure 1. This prototype was used to participate n the 2011 edition of the TAC english entity linking task.

4.1 Overview on the Approach

We specifically considered two simple query expansion mechanisms, namely one that finds acronyms for the named entity references by looking for a textual pattern that corresponds to having a set of capital words followed by the acronym inside parentheses (i.e., finding expressions like *General Electric (GE)*), or vice-versa, and another that looks for longer entity mentions in the source text (i.e., *SMART Communications* is an expansion for the query *SMART*). As for the candidate generation module, we considered an approach that returns the top-*k* most likely entries in the knowledge base, according to a modified version of the traditional cosine similarity computed between the query and all knowledge base entries. The modification essentially replaces the unit level, making the metric work with name n-grams instead of document words, with n between 1 and 4. Roughly, this means that the more n-grams the query string has in common with a name in the knowledge base, the more probable the respective entry is to being selected as candidate.

The candidate ranking module is based on supervised learning to rank approaches, given the success of previous works like those of Zheng et al. (2010) and He and de Rijke (2010). The highest ranked candidates are then filtered through a supervised classifier that detects the nil references.

Finally, the nil references are clustered together by applying the transitive closure to a graph with the nil queries as nodes, and with the initial edges estimated through a trained classifier that finds pairs of duplicate referents.

The rest of this section details the considered features for representing the association between named entity references and candidate entries, as well as the considered supervised learning models.

4.2 Considered Features

The considered learning methods for disambiguating entity references rely on a rich set of features, which can be organized according to the following groups.

4.2.1 Popularity Features

We considered a set of features that benefit more popular candidates, given the intuition that these candidates tend to be referenced more often in the texts.

- **Text Length Rank**. The rank of the given candidate in the list of all candidates, when they are ordered according to their textual description lengths.

- **Alternative Names Rank**. The rank of the given candidate in the list of all candidates, when candidates are ordered according to the corresponding number of alternative names.

4.2.2 Text-based Similarity

These features measure the similarity between the context where the entity reference occurs and the textual description for the candidate disambiguations.

- **Cosine Document Similarity**. The cosine similarity, using TF-IDF weights, between the candidate's description and the query source text, i.e., the document where the query occurs.

- **Cosine Near Context Similarity**. The cosine similarity, using TF-IDF weights, between the candidate's description and a window of 50 tokens surrounding all occurrences of the query.

- **Cosine Named Entity Similarity**. The cosine similarity, using TF-IDF weights, between the candidate's description and the query.

- **Cosine with Pseudo-Relevance Feedback**. The cosine similarity, using TF-IDF weights, between the candidate's description and the 20 words with the highest TF-IDF weights from the 5 entries in the entire knowledge base with a description most similar to the query's near context (i.e., the window of 50 tokens surrounding all query occurrences).

- **Cosine Document Rank**. The rank of the given candidate in the list of all candidates, when candidates are ordered according the feature which we called cosine document similarity.

- **Query in Candidate's Text**. One if the query occurs in the candidate's description, zero otherwise.

- **Candidate's Name in Source Text**. One if the candidate's main name occurs in the query's source text, zero otherwise.

4.2.3 Topical Similarity

This set of features leverages on topic-based representations for the query's source text and the candidate's description, as obtained through a Latent Dirichlet Allocation (LDA) topic model **?** built with basis on all descriptions in the knowledge base. By representing the source and candidate texts as probabilistic distributions over topics, as opposed to bags-of-words, we hope to minimize the impact of vocabulary mismatches. Our LDA topic model was generated with the contents of the knowledge base, pre-processed in order to remove stop-words and case information, and word terms reduced to their corresponding stems through Porter's algorithm. The actual model was built through a Gibbs sampling procedure with 200 iterations, using the 22,000 most frequent word stems. The values of the α and β hyper-parameters were respectively set to $50/K$ and 0.1, where $K = 200$ is the considered number of topics.

- **Topic Vector Similarity**. The cosine similarity, computed between the vectors corresponding to the candidate and the query's topic probabilities.

- **Topic Match**. One if the topic that best characterizes the candidate's description is the same that best characterizes the source text, zero otherwise.

- **Topic Divergence**. The symmetrized form of Kullback-Leibler divergence metric, computed between the candidate and query's topic distributions.

4.2.4 Name Similarity

These features capture similarities between the strings of the entity references and the candidate's names.

- **Exact Name Match**. One if the named entity is an exact match with at least one of the possible names for the specified candidate, zero otherwise.

- **Name Substring**. One if the entity name or one of the candidate's names is a substring of the other, zero otherwise.

- **Query Starts Candidate Name**. One if at least one of the candidate's names starts with the query, zero otherwise.

- **Query Ends Candidate Name**. One if at least one of the candidate's possible names ends with the query, zero otherwise.

- **Candidate's Name Starts Query**. One if the entity name starts with at least one of the candidate's names, zero otherwise.

- **Candidate's Name Ends Query**. One if the entity name ends with at least one of the candidate's names, zero otherwise.

- **Common Name Words**. The maximum number of common words between the query and one of the candidate's names.
- **Levenshtein Similarity**. The string similarity based on the Levenshtein metric between the candidate's main name and the query. Noticing that queries are often expanded with basis on the source document, we also check if the expanded query results in a higher feature similarity, using this value if it is indeed the case.
- **Jaro-Winkler Similarity**. The string similarity based on the Jaro-Winkler metric between the candidate's main name and the query or query expansion, whichever is higher.
- **Jaccard Similarity**. The string similarity based on the Jaccard token-based metric between the candidate's main name and the query .
- **Soft Jaccard Similarity**. The Jaccard token-based string similarity between the candidate's main name and the query, using the Levenshtein edit-distance with a threshold of 2 when matching individual tokens.
- **Soft TF-IDF Similarity**. The TF-IDF token-based string similarity between the candidate's main name and the query, using the Jaro-Winkler distance with a threshold of 0.9 when matching individual tokens.
- **Named Entity Comparison** This Named Entity Similarity Metric[1] applies sets of rules to two strings to determine whether they are likely to refer to the same underlying entity. It handles different types of entity – people, locations, and organizations – using appropriate resources (e.g., acronyms for companies).

4.2.5 Entity and Geo-based features

These features leverage on the results from a Named Entity Recognition (NER) system, specifically Stanford NER[2], applied to both the query's source text and the candidate's description. NER systems return not only the named entities occurring in the text but also their estimated type (i.e., person, organization, or location).

- **Common Entities**. The number of named entities shared by both the query's source text and the candidate's textual description.
- **Jaccard Similarity Between Entities**. The Jaccard similarity metric computed between the set of named entities in the query's source text and the set of named entities from the candidate's description.
- **Common Geo Entities**. The number of named entities of the location type that are shared by both the query's source text and the candidate's description.
- **Missed Geo Entities**. The number of location entities in the source text, but not in the candidate's description.
- **Query Type**. The named entity type, i.e. person, organization, location, or unknown, estimated when recognizing a named entity with the same string as the query. If the query is recognized more than once, the type results from a majority vote. Each type is represented by a binary feature.
- **Candidate Type**. Similar to the query type, but based on the candidate's description.
- **Type Match**. One if the query and the candidate's types are the same, zero otherwise.

4.2.6 Page-Type Features

These features provide information regarding the type of document where the query occurs.

- **Page Type**. The source type of each text associated with a query (e.g., Web, newswire). Each type is represented by a binary feature.

4.2.7 Validation-Only Features

Besides the aforementioned features, and noticing that the validation module only takes as input the top ranked candidate (i.e., the best possible disambiguation), we consider some features that result from the full set of ranking scores and check if the score for the top-ranked candidate is significantly different from that of the remaining candidates.

- **Ranking Score**. The score of the best candidate, as given by the ranking module.
- **Mean Score**. The mean score given to the query's set of candidates.
- **Difference from Mean Score**. The difference between the best candidate's ranking score and the mean score given to the query's set of candidates.
- **Standard Deviation**. The standard deviation in the scores given to the query's set of candidate.
- **Dixon's Q Test for Outliers**. This feature is obtained through the formula $(x_1 - x_2)/(x_1 - x_{last})$ where x_1 denotes the score of the top-ranked candidate, x_2 denotes the score of the second-ranked

[1] http://cogcomp.cs.illinois.edu/page/demo_view/26

[2] http://www-nlp.stanford.edu/ner/

candidate, and x_{last} denotes the score of the candidate ranked in last. If the first candidate is different from the others, then it is likely to correspond to an outlier.

- **Grubb's Test for Outliers**. This feature is obtained through the formula $(x_1 - avg)/stdev$, where x_1 denotes the score of the top-ranked candidate.

4.3 Supervised Learning Models

We experimented with five different state-of-the-art Learning to Rank (L2R) algorithms for building the candidate ranking model, namely the Ranking SVM (Joachims, 2002) and Ranking Perceptron (Collins and Duffy, 2002) pairwise L2R methods and the ListNet (Cao et al., 2007), Coordinate Ascent (Metzler and Bruce Croft, 2007) and AdaRank (Xu and Li, 2007) listwise methods.

Ranking SVM transforms the ranking problem into a set of binary classification tasks which are addressed through the formalism of Support Vector Machines (SVM). Ranking Perceptron is pairwise ranking model based on a variation of the well-known perceptron linear classifier.We essentially classify pairs of instances according to which one is more relevant to the query, and the goal is to minimize the number of misclassified pairs.

ListNet is a listwise approach which defines a ranking loss based on the probability distribution of the permutations. Coordinate Ascent is a state-of-the-art listwise method which uses the coordinate ascent unconstrained optimization technique to optimize the ranking model parameters. AdaRank is a listwise boosting technique for ranking, which directly maximizes any desired metric computed over ranked lists of candidates. The ranking model is essentially a linear combination of weak rankers, which can theoretically be of any type, although they are most commonly chosen as a binary function with a single feature and a threshold.

All ranking algorithms are available through the MinorThird[3], RankLib[4], and SVM$rank$[5] software libraries.

As for the candidate validation module, we experimented with an SVM classifier with a Radial Basis Function as kernel and also with a Random Forest classifier.

Each of these classifiers models the problem with different levels of complexity. For instance, Random Forest classifiers try to define a function that logically partitions the classification space in terms of a tree of decisions made over attributes of the original data, whereas SVMs use a kernel function to flexibly map the original data into a higher-dimensional space, where a separating hyper-plane can be defined.

SVMs exhibit a remarkable resistance to noise, handle correlated features well, and rely only on most informative training examples, which leads to a larger independence from the relative sizes of the sets of positive and negative examples. They are currently the most widely-used classification technique. Random Forests, on the other hand, are a more recently proposed ensemble method, consisting of many decision trees and combining bagging with the random selection of features. This are currently one of the best learning algorithms available, although Random Forests do not handle large numbers of irrelevant features as well as other methods.

All classification algorithms are available through the SVM$light$[6] and Weka[7] software libraries.

Following similar ideas to those of Peng et al. (2010) for document-retrieval, we also experimented with the usage of multiple learning models, each one trained for a specific type of query. The idea is to select, at run-time, an appropriate model for ranking or validating the candidates of the given query. Therefore, we considered having four model types readily available, namely models for people entities, models for locations, models for organizations and generic models for unknown entity types.

4.4 Resolving NIL Entities

When the validation module decides that the top-ranked candidate is not the correct disambiguation, we assume that the correct referent is not present in the knowledge base (i.e., we have a nil entity). Instead of assigning the reference to the corresponding knowledge base entry, nil entities should be disambiguated by clustering them together, i.e. by grouping the entity references that belong to a same referent, even-though we do not have its description in the knowledge base. The proposed approach for resolving nil entities involves the following steps:

1. Build a classification model for detecting duplicate entity references.
   (a) Find pairs of training queries with a string similarity greater or equal than a given threshold;
   (b) Compute all but the popularity-based ranking features for the training pairs;
   (c) Assign positive labels to query pairs referring to the same entity, and negative labels to the others;
   (d) Discard all query names without a positive and negative example;
   (e) Build a classification model.
2. Classify pairs of duplicate entity references.

[3] http://sourceforge.net/apps/trac/minorthird/
[4] http://www.cs.umass.edu/~vdang/ranklib.html
[5] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[6] http://svmlight.joachims.org/
[7] http://weka.wikispaces.com/

(a) Find all pairs of test queries with a positive nil estimate and a string similarity greater or equal than a given threshold;

(b) Compute all but the popularity-based ranking features for the testing pairs;

(c) Apply the model and build a query graph where query pairs with a positive estimate are connected;

3. Cluster references through the transitive closure.

Notice that the proposed approach essentially reuses the learning to rank method from the candidate ranking module, with query-query pairs instead of the query-candidate pairs.

5 Experimental Validation

We compared different configurations of the proposed entity linking approach using the training and testing datasets from previous TAC-KBP events Ji and Grishman (2011). Table 1 presents characterization statistics for the considered datasets. The queries correspond to named entities belonging to one of three groups, namely people, organizations, and geopolitical entities. The considered knowledge base was also provided by the TAC-KBP organization, and it consists of roughly 800.000 English articles from an October 2008 Wikipedia snapshot.

Unless otherwise stated, we use accuracy to measure system performance. Formally, this metric corresponds to the ratio between the number of correctly disambiguated queries, divided by the total number of queries.

CORPUS	NIL	PER	ORG	GPE	ALL
Train 2009	57.1%	627	2710	567	3904
Test 2009	28.4%	500	500	500	1500
Train 2010	49.1%	1127	3210	1067	5404
Test 2010	54.7%	750	750	750	2250
Train 2011	50.8%	1877	3960	1817	7654
Test 2011	50.0%	750	750	750	2250

Table 1: Number of entity mentions in the TAC-KBP datasets.

The rest of this section details the set of experiments which was designed to evaluate different aspects of our entity linking system.

5.1 Candidate Selection

A fundamental aspect in entity linking is the number of candidates that are passed to the ranking module. Selecting a high number of candidates would improve recall, lowering the number of cases when the correct referent exists in the knowledge base but is not selected for ranking. However, more candidates also means more feature computations, as well as more noise, since candidates with lower name similarities will be considered. Our experiments showed that, in our usual system setup, selecting up to 30 candidates for each query results in a considerably low candidate miss rate – as shown in Table 2. A manual analysis of the results showed that the majority of candidate misses comes from highly ambiguous place names (e.g., *Columbia* or *St. Louis*), followed by acronyms (e.g., *TNT*, *HDFC* or *SF*) and generic entities (e.g., *democratic party* or *public security police*).

Dataset	Misses	% of total queries
2009	52	3.5%
2010	47	2.1%
2011	92	4.1%

Table 2: Number of candidate misses.

Regarding the candidate selection process, the aforementioned simple query expansion techniques are very important to reduce the candidate misses and consequently improve system performance. In the most recent dataset (i.e., TAC-KBP 2011), the query expansion module decreases candidate misses by more than 50%, thus improving system accuracy by 4%.

5.2 Feature Contribution

One important and interesting question is the contribution of the different types of features. Namely, how important is a particular type of information to the named entity linking task. We studied this problem by removing features of a specific type to see how much they contribute to the final accuracy scores. Results are presented in Table 3, and they were obtained using our best performing system setup in the 2011 TAC-KBP dataset.

Features	PER	ORG	GPE	All
All	88.4%	78.0%	71.6%	79.3%
-Name Sim.	87.1%	75.1%	66.0%	76.0%
-Text Sim.	87.1%	74.7%	66.7%	76.1%
-NER	87.1%	75.7%	68.8%	77.2%
-LDA	87.6%	76.8%	68.5%	77.6%
-Popularity	86.9%	78.3%	68.8%	78.0%
-Page Type	88.1%	77.6%	71.5%	79.1%

Table 3: Accuracy after removing sets of features.

The results show that name and text similarity features are the most helpful, since the system performance dropped significantly after removing them. However, the impact of other features was not so clear. Given the small differences in performance after each type of features is removed, it is possible that several of those features are

complementary or redundant. To confirm it, further experiments were made, where instead of removing one type of feature from a complete system, we added the features from this type to a baseline system. We considered as baseline a system with just the text similarity features, which significantly outperformed all other options according to a separate experiment. Table 4 presents the obtained results. One of the most interesting conclusions is that feature importance greatly depends on the type of query, which further motivated us to perform a set of experiments with query-based models.

As a side note, regarding the LDA features, other experiments showed that changing the number of topics did not significantly influence the system performance.

Features	PER	ORG	GPE	All
Text Sim.	84.9%	73.5%	62.5%	73.6%
+NER	87.6%	76.7%	66.0%	76.8%
+Name Sim.	86.8%	73.1%	68.0%	76.0%
+Popularity	82.7%	75.3%	65.5%	74.5%
+LDA	86.5%	75.1%	61.3%	74.3%
+Page Type	83.3%	75.7%	63.2%	74.1%

Table 4: Accuracy after adding sets of features to a baseline.

5.3 Query-based vs. Single Ranking Models

Following recent proposals for document retrieval, and considering the observations from the previous experiments, we also wanted to see if using ranking and validation models specific to a given query type could improve results in our entity linking system. However, the results shown in Table 5 are not conclusive, leading us to believe that considering query-based models does not justify the increased complexity to the system.

In order to estimate the query type we used the classifications given by the Stanford NER (i.e., Person, Organization, Location) to the majority of named entities with the same string as the query. Table 6 shows the accuracy of our heuristic to determine query types.

5.4 Learning Algorithms

In the following experiment, we measured the impact that different ranking algorithms could have on the system. We also experimented with different validation algorithms, namely SVMs and Random Forests, but the later consistently outperformed the former, reason why we focus our presentation of the results on the ranking algorithms which produced more variability. All the results presented in the paper were produced with a random forest classifier as the algorithm in the validation module.

Our results show that there is no ranking model that clearly outperforms the others, although the Coordinate Ascent does obtain the best performance in two of the datasets. However, as a downside, it is the most time consuming algorithm among the considered group. Overall, the results for 2009 and 2010 are very competitive with the best reported results (82.2% and 85.8%, respectively).

Ranking Model	Dataset	Variation
SVMrank	2009	-1.4%
	2010	+1.4%
	2011	0.0%
Ranking Perceptron	2009	-2.0%
	2010	+1.3%
	2011	+0.6%
ListNet	2009	+0.2%
	2010	+1.1%
	2011	0.0%
Coordinate Ascent	2009	-3.3%
	2010	-0.6%
	2011	-1.2%
AdaRank	2009	+0.4%
	2010	+0.1%
	2011	-1.5%

Table 5: Results in terms of accuracy when using the query-based approach instead of a single model.

Dataset	PER	ORG	GPE	ALL
2009	90.6%	78.6%	92.6%	87.3%
2010	93.7%	69.9%	82.1%	81.9%
2011	85.5%	69.1%	83.9%	79.5%

Table 6: Accuracy when determining query types.

Ranking Model	Dataset	Accuracy
SVMrank	2009	81.7%
	2010	83.5%
	2011	**79.3**%
Ranking Perceptron	2009	81.7%
	2010	84.6%
	2011	79%
ListNet	2009	80.2%
	2010	83.2%
	2011	76.8%
Coordinate Ascent	2009	**82.3**%
	2010	**84.8**%
	2011	78.8%
AdaRank	2009	78.3%
	2010	83.3%
	2011	76.0%

Table 7: Results in terms of accuracy when using the different learning to rank algorithms.

The results for the 2011 dataset were weaker than we anticipated, 0.74, 0.78, and 0.76 in terms of B^3 precision, recall, and F_1, respectively.

The considerable drop in performance in the 2011 dataset might be explained by two main reasons: (i) we have a considerably high performance with nil queries and these are almost 5% less than in 2010, and (ii) the queries seem more difficult than in past years. Supporting this last point is our observation that, compared to 2010, there was a large drop in the usefulness of text similarity, name similarity, and popularity features, suggesting that the query's context and name are more ambiguous and that less queries correspond to the most well-known entity answering by that query name.

A more detailed analysis of our best system configuration for the 2011 dataset is shown in Table 8. The results show that the disambiguation of geo-political entities seems particularly challenging, with the system achieving the worse results on this particular entity type. Notice that the accuracy reported for the ranking module takes only into consideration non-nil queries, and the reported validation accuracy ignores the queries which were incorrectly ranked.

Module	PER	ORG	GPE	ALL
Ranking	87.6%	71.9%	77.0%	77.9%
Validation	92.5%	89.3%	85.2%	89.2%
All	88.4%	78.0%	71.6%	79.3%

Table 8: Detailed results for our best system in the 2011 dataset.

6 Conclusions and Future Work

Despite the recent advances in the named entity linking task, we have that approaches based on machine learning have not been carefully discussed in the literature, leaving several open questions for those trying to develop such systems. Through this work we have presented and thoroughly evaluated a relatively simple learning-based approach for named entity disambiguation, which uses a rich set of features and out-of-the-box learning to rank methods to achieve a performance in line with that of current state-of-the-art approaches.

Our experiments confirm recently published results, which show that machine learning methods with relatively simple to compute features can match the current state-of-the-art. Furthermore, we showed that selecting the most suitable learning to rank model with basis on the query type brings little improvements to the results. For future work, we intend to experiment with machine learning models that use different sets of features better tailored for each type of entity, particularly for the case of geo-political entities where we hope to be able to use features based on geospatial properties (e.g., the spatial proximity towards other geo-political entities referenced in the same context).

For future work, it would also be interesting to explore the usage of more advanced query expansion mechanisms (e.g., consider hypertext anchors from Wikipedia links as alternative names for the knowledge base), as well as the usage of additional features. The considered set of features does not, for instance, consider linkage information from Wikipedia, while some previous works have leveraged on the number of inlinks, or on algorithms such as PageRank, as a way of estimating candidate importance. We believe that features derived from structured information associated to the knowledge based entries (i.e., features derived from slot-filling methods) could provide rich information for entity disambiguation purposes.

The Information Retrieval community has also started to look at the problem of relational learning to rank, explicitly considering cases in which there exists relationship between the objects to be ranked (Qin et al., 2008). For future work, and noticing that entities referenced in the same context should be similar to one another, we would like to experiment with such methods in order to explore full-document disambiguations.

Acknowledgements

This work was partially supported by the Fundação para a Ciência e a Tecnologia, through project grants with references PTDC/EIA-EIA/109840/2009 (SInteliGIS) and PTDC/EIA-EIA/115346/2009 (SMARTIES), and through the PhD scholarship SFRH/BD/71163/2010.

We would also like to thank David Matos and Mário J. Silva for their assistance and insightful comments.

References

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc.

Bagga, A. and Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics - Volume 1*, COLING '98, pages 79–85. Association Computational Linguistics.

Bunescu, R. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the European Conference of the Association for Computational Linguistic*, EACL '06.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136. ACM.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, pages 708–716.

Etzioni, O., Banko, M., Soderland, S., and Weld, D. S. (2008). Open information extraction from the web. *Commun. ACM*, 51:68–74.

Gale, W. A., Church, K. W., and Yarowsky, D. (1992). One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, HLT '91, pages 233–237. Association for Computational Linguistics.

Geng, X., Liu, T.-Y., Qin, T., Arnold, A., Li, H., and Shum, H.-Y. (2008). Query dependent ranking using k-nearest neighbor. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 115–122. ACM.

He, J. and de Rijke, M. (2010). A ranking approach to target detection for automatic link generation. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 831–832. ACM.

Ji, H. and Grishman, R. (2011). Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142. ACM.

Leidner, J. (2007). *Toponym Resolution: a Comparison and Taxonomy of Heuristics and Methods*. PhD thesis, PhD Thesis, University of Edinburgh.

Li, H. (2011). Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113.

Li, P., Burges, C., Wu, Q., Platt, J., Koller, D., Singer, Y., and Roweis, S. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in Neural Information Processing Systems*.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Metzler, D. and Bruce Croft, W. (2007). Linear feature-based models for information retrieval. *Inf. Retr.*, 10:257–274.

Mihalcea, R. and Csomai, A. (2007). Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242. ACM.

Mohan, A., Chen, Z., and Weinberger, K. Q. (2011). Web-search ranking with initialized gradient boosted regression trees. *Journal of Machine Learning Research, Workshop and Conference Proceedings*, 14:77–89.

Peng, J., Macdonald, C., and Ounis, I. (2010). Learning to select a ranking function. In *32nd European Conference on Information Retrieval*, ECIR '10, pages 114–126. Springer.

Qin, T., Liu, T.-Y., Zhang, X.-D., Wang, D.-S., Xiong, W.-Y., and Li, H. (2008). Learning to rank relational objects and its application to web search. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 407–416. ACM.

Sarmento, L., Kehlenbeck, A., Oliveira, E., and Ungar, L. (2009). An Approach to Web-Scale Named-Entity Disambiguation. In *Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM '09, pages 689–703. Springer-Verlag.

Whitelaw, C., Kehlenbeck, A., Petrovic, N., and Ungar, L. (2008). Web-scale named entity recognition. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 123–132. ACM.

Xu, J. and Li, H. (2007). Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 391–398. ACM.

Zheng, Z., Li, F., Huang, M., and Zhu, X. (2010). Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 483–491. Association for Computational Linguistics.

Zhu, Z. A., Chen, W., Wan, T., Zhu, C., Wang, G., and Chen, Z. (2009). To divide and conquer search ranking by learning query difficulty. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1883–1886. ACM.