Combining similarities with regression based classifiers for Entity Linking at TAC 2010

César de Pablo-Sánchez, Juan Perea, Paloma Martínez

{cdepablo,jiperea,pmf}@inf.uc3m.es
Computer Science Department
Universidad Carlos III de Madrid
28911 Leganés, Spain

Abstract

The UC3M team has sent three runs for each Entity Linking task proposed in the Knowledge Base Population (KBP) track at TAC 2010. The skeleton system presented in 2009 has evolved in 2010 by incorporating some new tools, new algorithms for candidate retrieval and feature extraction, and two new stages that use regression based classifiers for candidate filtering. These improvements have allowed the UC3M team overall values to almost reach the median values of all participants in the Entity Linking tasks.

1 Introduction

The Knowledge Base Population task aims at advancing the state of the art for systems that automatically populate the Knowledge Base (KB) of an ontology with facts about their instances. The task focuses on acquiring knowledge for Named Entities (NE) from a large document collection and is further divided in two key tasks, Slot Filling and Entity Linking. The goal of Slot Filling is to update the KB with new information extracted from the collection given a set of important attributes (or slots) for a query that is a person or an organization. The second task, Entity Linking is aimed at solving the problem of finding the correct entity to update given a mention in context. The task includes particularly difficult cases like ambiguous mentions (e.g *George Bush*), aliases (e.g *Angela Kasner*, more known as *Angela Merkel*) or even examples of both (e.g *ABC*).

This year, our team (UC3M) focused only on the second task, Entity Linking, and made significant improvements to our last year sys-

Type		Count	Percentage
Person	PER	114.523	14.0%
Organization	ORG	55.813	6.8%
Geo Political Entity	GPE	116.499	14.2%
Unknown	UNK	531.907	65.0%
All		818.741	

Table 1: KB nodes by entity types

tem. Our approach is based on indexing information from the KB to find a list of candidates that includes a correct link for every mention with high recall. In sucessive steps the set of candidates is filtered using contextual and name similarity clues to produce a single candidate or NIL if no adequate candidate is found. The filters are implemented as regression based machine learning classifiers that combine the different similarity metrics and numeric scores to produce a final ranked list of candidates with their scores.

2 The Task: Entity Linking

The Entity Linking task assumes that we have a large list of entities organized in a Knowledge Base (KB) and a large document corpus. In TAC2010 the KB is semi-automatically derived from Wikipedia. It is composed by four types of entities: persons (PER), organizations (ORG), geo-political entities (GPE) and those whose type is unknown (UNK). Each entity in the knowledge base contains title, name, type, id, some wiki text, and several facts of different types in the form of a [name, value] pair. Table 1 summarizes data from the KB as presented in [4].

The document collection is composed of two different parts, a 1.3 million English newswire articles and 488.240 webpages. Newswire documents were published between 1994 and 2008 and were used also TAC 2009. In contrast, webpages have been introduced this year.

The evaluation is performed with a list of queries composed by [name-string, document, type]. Each name-string is mentioned in the document and may refer to one of the entities of the KB . An Entity Linking system will have to determine, for each of the queries, which of the entities in the KB, if any, is being referred to by the name-string in the given document. If the KB does not contain the referred entity then the correct answer is `NIL` to indicate that it does not appear. So far in TAC 2010, it is not required "create" a new KB entity that groups all the mentions or name-strings that co-refer but do not have

	Eval Data 2009 GS2009	Training Data 2010 TR2010	Eval Data 2010 GS2010
document genre	news	web	news + web
# queries	3904	1500	2250
non-NIL queries	1675	1074	1020
NIL queries	2229	426	1230

Table 2: Data sets outline

an entity in the KB.

There are several datasets available for training the system, the gold standard from TAC 2009 has been complemented with additional training data in 2010 created by LDC which used web documents. Table 2 summarizes the characteristics of each dataset including the evaluation data for TAC 2010. Systems are compared using Accuracy (micro-Accuracy) in their prediction, usually breaking down for non-NIL queries and NIL queries.

Besides, an optional Entity Linking task has been enabled this yeat which aims at building systems that do not use the wikitext available for KB entities, as large document text are rare in lots of practical KBs.

3 System description

The UC3M system for entity linking is composed of three main stages that work online: candidate entity retrieval, candidate filtering and NIL classification. It is based on the combination of several similarity measures and scores using logistic regression. Additional processing is required for indexing, scoring and normalization to create the features for each instance or candidate pair. Figure 1 depicts the main components of the system and the required offline processing.

The candidate retrieval stage adresses the first problem in the entity linking task, matching mentions to KB entities despite not being mentioned with the same name. Acronyms, aliases, name variants and typographical errors are among the reasons to expand the candidate list beyond exact string matchings.

The goal of the second stage is to reduce the size of a potentially large candidate list. It helps to perform the second problem in EL, disambiguation. We have cast the problem as a simple classification problem to decide if a pair (query,entity node) is correct. We integrate similarities between the contexts available for the query, its document, and the KB entity: the wiki document, the infobox and the article

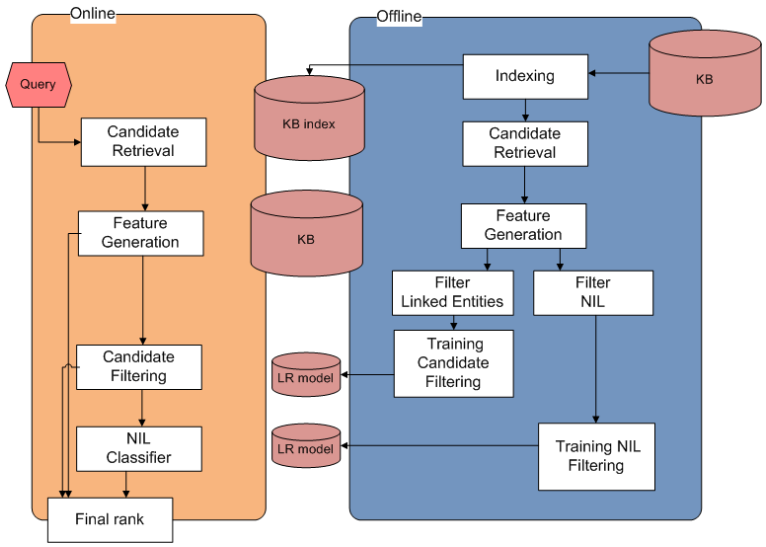

Figure 1: Entity Linking architecture

links. A classifier may provide several links to the KB and the probability of the classification is used to rank them. On the other hand if no pair is identified as a link the final answer is NIL. The third step is another classifier but, in this case, it is specialized in detecting if the link between the pair is correct or not.

We have used a similar architecture for the main EL task and the optional task although details on how training data is generated and features differ. We have test the use of different training sets for the main EL task. In the optional task we introduced additional name similarities as features for the two classifiers.

3.1 Indexing

Before any query can be run, the knowledge base entities and the article corpus need to be indexed using Lucene [1]. For each query, it can return a scored list of matching documents, so it fits our needs quite nicely. Searches are executed only against the KB index; the article corpus index is created solely for storage purposes, as locating an article in the file system could be quite tedious given the amount of directories.

Four indices are created from the knowledge base: the KB index, the ALIAS index, the NER index and the WIKIPEDIA index. For

[1] http://lucene.apache.org

each entity in the knowledge base, a document is created containing the following fields:

- For all indices: *id* and *wikiTitle* are extracted from the supplied knowledge base.
- For the KB index: *name*, *type*, *wikiText* and *factValueList*. This information is extracted directly from the supplied knowledge base. This index is used only to get a fast access to entity data.
- For the ALIAS index: *nameList*, *acronymList* and *domainList*. This information is extracted with no or little processing from facts in the KB, and each field can contain different names that can be used to reference the given entity. For example, facts *alias*, *abbreviation* and *website* will be used to populate *nameList*, *acronymList* and *domainList* respectively with no processing at all, and fact *name* can be used to populate *acronymList* with some processing. This index will be used during the candidate retrieval stage.
- For the NER index: only the *namedEntityNameList* field is appended. This field contains the NE found by the Stanford Named Entity Recognizer [2] [**?**] in the knowledge base text. Named entities found will be matched against named entities found in the articles.
- For the WIKIPEDIA index: fields *anchorList*, *categoryList*, *redirectList*, *outLinkList* and *inLinkList* are appended if the corresponding record is found in the Wikipedia snapshot. Information in this index can be used to complement information in the ALIAS and NER indices.

3.2 Candidate retrieval

For an entity to be a candidate, it needs to have some similarity in any relevant field with the name-string in the query. This similarity could also be with any synonym, alias or acronym extracted from the name-string itself or from the document text. The candidate list is obtained by running simple Lucene searches on the search fields of the ALIAS index (nameList, acronymList and domainList). Each of this searches returns a list of [document, score] pair, each representing a candidate entity. Lucene can return huge document lists for each search, so we need to filter candidate lists to save processing time. A threshold score, which can be different for each of the three searches, is defined based on the training data. Scores for the correct answers are sorted, and we choose a threshold that would let most of these correct answers

[2] http://nlp.stanford.edu/ner/index.shtml

in the candidate list. Entities in each list with a score over the given threshold are merged into an overall list.

3.3 Feature Extraction

Once a candidate list for a given [entity name, document] query has been populated, we need to choose which of the candidates in the list is the best option. During the candidate retrieval stage, we got some scores (**index-based features**) that might show how good each candidate is.

However, these scores are only related to the entity name in the query, and more numerical features need to be calculated to show the correlation between each candidate and the document text. For example, it is expected that some values in the *factValueList* field and some names in the *namedEntityNameList* field (names that appear in the knowledge base text) will also be present in the document text. The system implements two algorithms that extract some **context similarity features**:

- Lucene search using the document text - First, a single-document Lucene index is built using the document text. Then, given some data from the candidate, a simple search is constructed and run against such index, and the resulting score is stored as a feature.
- Cosine distance - Given some data from the candidate, the cosine distance is used to get a value that shows how much this data matches the document text.

Data used to feed these algorithms include the following fields in the knowledge base indices: *nameList*, *acronymList*, *domainList*, *factValueList* and *namedEntityNameList*. Finally, once the candidate list has been scored we produce for every of the above scores two values, a raw one and a normalized one. The last one is normalized with the highest value in the candidate list.

We also included a number of **name similarity features** in some of the experiments for the optional task. Though index-based features measure name similarity, they are global measures. For instance, they take TF.IDF weights into account to downweight common terms like *George* in comparison to uncommon ones like *Bush*. We used simple similarity measures and some string similarity measures provided in the SecondString [3] package between the query and the entity title . We used **equal**, **QcontainsE** (the entity is a substring of the query), **EcontainsQ**, **Jaccard**, **Jaro**, **JaroWinkler**, **MongeElkan** and **SLIM**.

[3] http://secondstring.sourceforge.net/

3.4 Candidate Filtering

The Candidate Retrieval stage provides a large set of candidate pairs (query,KB entity) but on the optimal situation only one of them should be selected as the correct link. We have relaxed this problem by modelling it as a classification problem that aims at filtering those pairs that are not related taking into account contextual features. The classifier may decide that several pairs are good candidates and then the prediction confidence is used to rank the rest of candidates.

Training sets were generated with the subset of queries that were linked to a KB entity (non-NIL). For each query, a candidate list is retrieved and a vector of features generated for each pair. An obvious problem with this approach is that the datasets are highly imbalanced. At best, there is one correct pair in a list of dozens or even hundred candidates. We have attempted to overcome this problem by using cost-sensitive classifiers. Our cost matrix assigns 10 times higher cost to filter out a correct candidate (false negative) than to let an incorrect one pass (false positive). We seek here to maximize the recall and leave part of the work for the next step, the NIL classifier.

On the other hand, an advantage of this approach is that we can use several standards classifiers like those provided by Weka [4] [5]. We tested several classifiers that used to perform well with numeric values like Logistic Regression and Support Vector Machines but found during system development that the Classification via Regression scheme [2] outperformed both in terms of Recall with similar Precision. The Classification via Regression algorithm uses model trees, decision trees with linear regression functions at the leaves, as the basis for classification.

During the experiments we found that the document (web or news) had a larger influence on the results, specially when tested in cross-genre comparisons. Our three main-task runs use different training sets with the Classification via Regression scheme.

3.5 NIL classification

In contrast, the last classifier is aimed at detecting with high precision candidate pairs that are incorrect and therefore should not be linked to the KB. In this module, we have used a Logistic Regression classifier trained with a different subset of the queries. Positive examples were extracted from candidate pairs that are correct and found. Negative examples were create from the top ranked entities for those queries that do not have a link in the KB (NIL queries). Although the set is not perfectly balanced there is no such large as skew as the candidate filter classifier.

[4] http://www.cs.waikato.ac.nz/ ml/weka/

	UC3M1	UC3M2	UC3M3
2250 queries	0.6742	0.6507	0.6742
1020 non-NIL	0.5127	0.5873	0.4941
1230 NIL	0.8081	0.7033	0.8236

Table 3: Main task results

	UC3M1	UC3M2	UC3M3	Highest	Median
750 ORG	0.6867	0.6653	0.6667	0.8520	0.6767
749 GPE	0.5180	0.5287	0.5087	0.7957	0.5975
751 PER	0.8176	0.7577	0.8469	0.9601	0.8449
2250 ALL	0.6742	0.6507	0.6742	0.8680	0.6836

Table 4: Main task results breakdown by NE class

4 Results

Our team submitted three runs for each of the tasks. Runs for the main Entity Linking task have focused on the use of training data from different genres. In the UC3M1 run we used both datasets (GS2009+TR2010) while the other runs used only dataset from a single genre. Run UC3M2 used web dats (TR2010) and UC3M3 used only news data (GS2009).

Table 3 summarizes the results for these three runs that obtain quite similar overall micro-Accuracy. The first and the third runs have similar performance, with higher results for correctly assigning NIL queries than detecting correct links. Run UC3M2, trained only with web data, obtained higher accuracy for identifying correct links but also is more prone to provide spurious ones. Candidate retrieval is exactly the same for the three runs, which indicates that the second run achieves better recall on correct links. In comparison, our results are slightly below the median value (0.6836) but much lower than the highest accuracy (0.8680).

Linking results vary widely across different NE classes as shown in Table 4. PER are the class with less linkable entities but on the other hand they are considerably easier to link than GPE and ORG. It is also clear that the advantage of the UC3M2 is due to their better performance disambiguating non-NIL GPE entities and ORG entities.

	UC3M1	UC3M2	UC3M3
2250 queries	0.5844	0.6622	0.6978
1020 non-NIL	0.3520	0.3980	0.4735
1230 NIL	0.7772	0.8813	0.8837

Table 5: Optional EL task results

	UC3M1	UC3M1 nowikitext	difference UC3M1	UC3M3 nowikitext
2250 queries	0,6742	0,5844	0,0898	0.6978
1020 non-NIL	0,5127	0,3520	0,1607	0.4735
1230 NIL	0,8081	0,7772	0,0309	0.8837

Table 6: Comparison of main and pilot runs

4.1 Optional task: Entity Linking without text descriptions

The first run submitted for the pilot task (UC3M1) is equivalent to its main task counterpart but all features that use the wiki text have been removed. In contrast, the second and third run try to assess the usefulness of additional name similarity features that we added during the last week before the evaluation. UC3M2 includes the name similarity features only in the NIL classifier while UC3M3 includes them in both classifiers.

An outline of the three pilot runs is depicted in Table 5 and clearly shows that the name similarity features help to improve results in both runs. The NIL classifier on the second run reduces the number of incorrect links improves candidate ranking as it is deduced from the increase of non-NIL and NIL queries accuracy. Furthermore, adding name similarity measures to candidate filtering also produces a better filtering and reranking of candidates.

The analysis of the result by NE type provide a similar conclusion, GPE are the more ambiguous type and therefore the most difficult to link. Figure 2 shows that our name similarity features are specially helpful for the GPE and PER queries. Nevertheless, these features are useful in both classifiers for GPE queries, while for PER queries they provide the major improvement when added to the Candidate Filter.

Finally, Table 6 compares the two first runs of both tasks that are similar except for the removal of features based on the similarity of wikitext. As expected, the textual context have a great influence disambiguating entities as reflected by the loss in non-NIL accuracy.

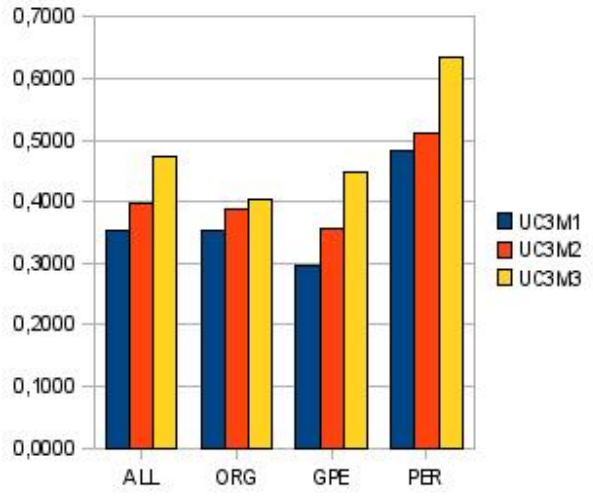

Figure 2: Results for non-NIL queries in the optional EL task

A similar decrease is shown across different types. However, when we add name similarity measures we achieve accuracy results similar to the main task, so we probably may expect to have a similar improvement by adding such features for main task classifiers.

5 Conclusion and Future Work

The UC3M Entity Linking system has combined a number of numeric scores and similarity metrics using machine learning classifiers based on regression. The system performs candidate retrieval on title, aliases ad domains exrtracted from the KB to find a large number of candidate pairs. Those are filtered based on contextual clues and finally validated by a NIL classifier. The goal is to achieve first high recall and then by filtering in succesive stage to achieve the desired precision. Runs were submitted for the main task and the pilot task and achieved results around the median performance. Good results were achieved for PER queries while the performance is lower for GPE and ORG queries. A direction for further research consists on the specialization of the classifiers in different NE types as has been carried in [3]. Most of our experiments have focused on the last stage classifiers and further research is required on the best way to integrate the retrieval module.

Besides, there are a large number of additional features that would work in our approach like a-priori information on linking [1] or trying

to perform collective entity linking for all the mentions of a document. Finally, the issue of document genre have shown to be important in our experiments and requires also detailed analysis of our results.

6 Acknowledgements

This work has been partially supported by the Regional Government of Madrid by means of the Research Network MAVIR2CM (S2009/TIC-1542) and by the Spanish Ministry of Education by means of the project BRAVO (TIN2007-67407-C3-01)

References

[1] Eneko Agirre, Angel X Chang, Daniel S Jurafsky, Christopher D Manning, Valentin I Spitkovsky, and Eric Yeh. Stanford-UBC at TAC-KBP. In *TAC2009 Working Notes*, Gaithersburg, 2009.

[2] Eibe Frank, Yong Wang, Stuart Inglis, Geoffrey Holmes, and Ian H. Witten. Using Model Trees for Classification. *Machine Learning*, 32(1):63, 1998.

[3] Fangtao Li, Zhicheng Zheng, Fan Bu, Yang Tang, Xiaoyan Zhu, and Minlie Huang. THU QUANTA at TAC 2009 KBP and RTE Track. In *TAC 2009 Working Notes*, Gaithersburg, 2009.

[4] Paul McNamee and Hoa Dang. Overview of the TAC 2009 Knowledge Base Population Track (DRAFT). In *TAC 2009 Working Notes*, number Ldc, Gaithersburg, 2009.

[5] Ian H Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.