UGent Participation in the TAC 2012 Entity-Linking Task

Laurent Mertens, Thomas Demeester, Johannes Deleu, Piet Demeester, Chris Develder
INTEC - IBCN
Ghent University, Ghent, Belgium
firstname.surname@intec.ugent.be

Abstract

This article describes in detail the system used by the UGent-IBCN team for participating in the Text Analysis Conference (TAC) 2012 Mono-Lingual Entity-Linking task. The presented system is essentially rule-based, following a generic framework that is highly optimised for each label (i.e. with different rules for persons, organisations, and locations). The main contribution of this work is in identifying a number of label-specific issues and presenting simple heuristic solutions that yet allow building an efficient and effective system. These treated issues include resolving abbreviated organisation names, resolving popular nicknames, or taking into account American vs British spelling.

1 Introduction

We approached the TAC 2012 English Entity-Linking task as a Named Entity Disambiguation problem (NED). The system we devised consists first of a Named Entity Recognition (NER) part, which we relegate to an external NER engine, followed by an NED part, in which we try to link, if possible, the recognised named entities to corresponding entities in the TAC Knowledge Base (KB).

Our system can be subdivided into four distinct phases. Starting from the TAC KB and the TAC article collection, we:

1. Extract specific information from the KB.
2. Apply NER to the articles.
3. Extract specific information from the NER output.
4. Apply our entity-linking system to the dataset, using all the information gathered in the previous steps.

Given a mention, we look for valid candidates in the TAC KB mainly using well-chosen string comparisons with known surface forms for the KB entries. The subsequent scoring of these candidates is done mainly through the determination of the textual overlap between an article and the candidate's description in the KB, looking for known facts about a candidate that also appear in the article, and looking if the entities referring to, or referred to by this candidate in the KB, also figure in the article. Often though, a mention is a derivation, or alternate form, of an entity's real name. We focused on a few specific cases: abbreviations of organisation names, nicknames and subtle differences between US and UK spelling.

After a brief overview of related work, each of the steps of our system is described in detail, and its effectiveness is illustrated with several experiments. The paper ends with some conclusions and notes on future research.

2 Related Work

Entity-Linking is the task of determining whether a named entity (e.g. a person, organisation or location) mentioned in a text refers to an existent entry in an external knowledge base (e.g. an encyclopedia). This task is a natural evolution of the Coreference Resolution problem, which has as goal to determine whether two or more mentions in a same document refer to the same physical entity, without necessar-

ily providing any link to (any representation of) said physical entity (Soon et al., 2001).

Cross-document Coreference Resolution is this same problem applied to mentions accros different documents. (Bagga & Baldwin, 1998) noted that this problem differs from the within-document case because one can not expect different documents containing identical mentions to be coherent, and because the linguistics related problems for within-document coreference get amplified when breaking the document boundary. Their approach is based on a Vector Space Model (VSM), where the comparison of mention-specific summaries for each document are used to decide upon the link between entities. Their work was further expanded by (Gooi & Allan, 2004), who targeted larger corpora.

When combining Entity-Linking with Cross-document Coreference Resolution, one gets Named Entity Disambiguation, the task of determining across a corpus of documents which mentions refer to the same physical entity, and whether this entity is present in an external KB, taking into account that identical mentions may refer to different entities (e.g. people having identical names). The most popular such KB in recent years has been Wikipedia. The first attempt at exploiting Wikipedia for NED has been performed by (Bunescu & Pasca, 2006), who used Wikipedia to gather surface forms for relevant entities, and using the immediate context of mentions of these entities in Wikipedia pages, as well as relations between context words and entity categories to disambiguate mentions.(Cucerzan, 2007) built a similar system, that also exploited Wikipedia list-pages to generate extra category tags, and used the Wikipedia linking structure to generate contexts for entities. Furthermore, they performed a within-document coreference step first mapping short surface forms to longer ones, before disambiguating the longest form in the thus generated clusters.

(Han & Zhao, 2009) stepped away from the ubiquitous BOW-models by using Wikipedia to generate not only a set of concepts for each relevant entity in Wikipedia through usage of Wikilinks, but also Semantic Relatedness between these concepts, based on work by (Milne & Witten, 2008).(Gentile et al., 2009) proposed a graph-based approach to NED, obtaining similar results as the VSM of (Cucerzan, 2007). For each surface form in a given text, they query Wikipedia, and take each listed page on the returned disambiguation page as possible candidate. They use this page to construct a feature space for each candidate. They create a weighted undirected graph using all candidates and their features as nodes, effectively connecting candidates sharing identical features. Using a random-walk algorithm they populate a relatedness matrix for the different candidates. NED is performed by mapping mentions on a single entity exploiting the information in this matrix. (Hoffart et al., 2011) exploited YAGO and DBpedia. They query these KBs to get a set of entities for each mention, then create an undirected graph with all mentions (M) and entities (E) as nodes. The ME-edges are weighted according to similarity and popularity prior measures, while the EE-edges are weighted according to Semantic Relatedness, also as defined by (Milne & Witten, 2008). Their goal is to compute a dense subgraph containing all mentions, and one entity-node per mention, that maximises all used measures.

An alternative departure from word-comparison techniques has been investigated by (Pilz & Paass, 2011), who use a topic-model using Latent Dirichlet Allocation to represent document context and entities.

3 Data Preparation

A large part of the data preparation consists in applying NER to the articles, and partially to the KB. For this, we used the Stanford NER system (Finkel et al., 2005), mainly because of the flexibility it offered as far as implemenation in our own Java code went.

3.1 Processing the TAC KB

The TAC KB contains a set of entities derived from about 800.000 Wikipedia pages, originating from an October 2008 snapshot. Such an entry consists of data gathered by the automatic parsing of the infoboxes from the original Wikipedia page, as well as a stripped version of the text of the Wikipedia article. An example of an entry from the TAC KB can be seen in Figure 1.

Each entry starts with a line containing the title of the Wikipedia page from which the information was

```
<entity wiki_title="Mike_Quigley_(footballer)" type="PER" id="E0000001" name="Mike
Quigley (footballer)">
<facts class="Infobox Football biography">
<fact name="playername">Mike Quigley</fact>
<fact name="fullname">Michael Anthony Joseph Quigley</fact>
<fact name="dateofbirth">October 2, 1970 (1970-10-02) (age38)</fact>
<fact name="cityofbirth"><link entity_id="E0467057">Manchester</link></fact>
...
</facts>
<wiki_text><![CDATA[Mike Quigley (footballer)

Mike Quigley (born 2 October 1970) is an English football midfielder.
]]></wiki_text>
</entity>
```

Fig. 1: Excerpt from the TAC KB

gathered, the label (denoted as "type") of the entry, a unique ID for the TAC KB, and the name of the entry (often the same as the Wikipedia page title). This is then followed by a listing of "facts", which were automatically parsed from the Wikipedia infobox and consist of a factname, and a factvalue. We refer to these as "WikiFacts". Finally, the entry concludes with a field containing a stripped version of the text of the Wikipedia article, the "WikiText". The possible label values are GPE (Geo-political Entity), ORG(anisation), PER(son) and UKN (unknown). Note the difference with the "typical" NER labels LOC(ation), MISC(ellaneous), ORG(anisation) and (PER)son.

From the KB, we extract a number of separate datasets, which are schematically depicted in Figure 2. We will proceed with explaining what these different datasets contain exactly, and later in the article we will explain how we use them to resolve mentions to KB entries. The main idea behind the pre-processing of the KB was not so much to create new information, but to bundle and concisely store different aspects of the data in order to streamline the workflow later on.

3.1.1 Extracting Surcace Forms from the KB

An essential part of our entity-linking approach is having an exhaustive list of surface forms for KB entries, so as to increase chances of recognising mentions, be it as full names, or as substrings of (surface forms of) KB entries. For this, we generated specific surface forms, using formatting rules, starting

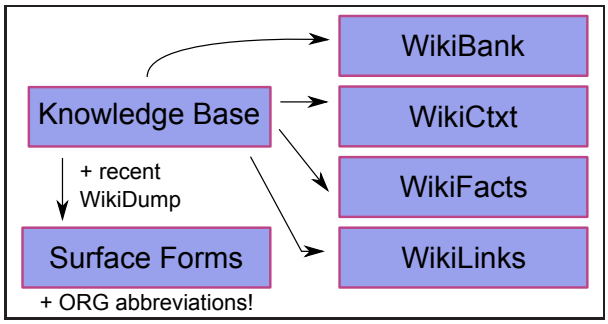

Fig. 2: Elements extracted from the TAC KB

from an entry's canonical name. Part of these rules consisted of splitting parts inbetween parentheses or following comma's (e.g. "`Cuba Gooding, Jr.`" becomes "`Cuba Gooding`"), generating lowercase forms, removing abbreviated parts (e.g. "`L. Ron Hubbard`" becomes "`Ron Hubbard`"), etc. Many organisations' canonical name, as it figures in the KB, ends on typical abbreviations like e.g. "corp.", "ltd.", etc. Were this was the case, we also generated a form without this last part. For example, for "Coca-Cola Corp." we generate a surface form "Coca-Cola".

In the same vein, we also used a list of common words whose spelling differs between American and British English[1]. If such words were found, we would generate a surface form using the equivalent from the other spelling. For example, "CUNY Graduate Center" leads to "CUNY Graduate Centre".

[1] http://www.wordsworldwide.co.uk/docs/Words-Worldwide-Word-list-UK-US-2009.doc

Some KB entries have an "abbreviation" or "acronym" fact. Where this was the case, we used the corresponding factvalues as surface forms. Specifically for organisations, we also used an external file[2] containing many common abbreviations for well known organisations to further increase the number of known abbreviations.

Apart from the TAC KB, we also used a recent Wikipedia dump (June 2012) to extract extra surface forms in many cases, using Wikipedia's redirects. For this, we checked the Wikipedia redirects for all TAC KB entries. In some cases, pages would be redirected to the KB entry, but in other cases, given that the Wikipedia dump was more recent, the original KB entry page would no longer exist, but be redirected to a newer page. The idea is simply to cluster all pages that are connected through redirects, and if this cluster contains an entity from the TAC KB, to add all the other (page)names as surface forms for this entity. This allowed us e.g. to add "Chris Breezy" as a surface form for "Chris Brown (entertainer)". This also proved to be another very fruitful source for finding abbreviations for organisations.

Furthermore, we also used keywords to filter out certain entities on the basis of their "disamb" feature, that proved to be needlessly increasing ambiguity (e.g. music albums, books, movies...). If an entity's canonical name ends with a part between parentheses, we refer to this part as the "disamb" feature, since often this part is meant to disambiguate between different entities with highly similar names. E.g. `E0800132 Elvis Has Left the Building (film)` is ignored on the basis that its disambiguation feature contains, and in this case even equals, `film`.

```
E0599293  Michael A. Jackson (sheriff)
#disamb sheriff
#form  Michael A. Jackson
#form  Michael Jackson
#form  Sheriff Michael A. Jackson
#form  michael a.  jackson
#form  michaela.jackson
```

Fig. 3: Example of surface form generation

[2] http://www.betweenthelakes.com/helpful/Abbreviations_for_organizations.htm

3.1.2 WikiBank

We did not only apply NER to the articles, but also to the WikiText fields in the KB. From this, we created the "WikiBank": a dataset which contains all named entities from the WikiTexts, as tagged by the NER system, per KB entry.

As an example, consider again the KB entry depicted in Figure 1. The corresponding WikiBank entry for this KB entry is shown in Figure 4.

```
#entity E0000001  PER  Mike Quigley
PER     Mike Quigley
MISC    English
```

Fig. 4: Excerpt from the TAC KB

3.1.3 WikiCtxt

The WikiCtxt is a dataset of tokens that we expect to be entity-specific. To obtain this list of tokens, all articles are parsed twice. During the first run, all words in all WikiTexts are lowercased, and all special characters (brackets, punctuation marks, ...) are removed. For all these normalised words, we count the number of times they appear in the total dataset, as well as in how many different documents they appear. Tokens that contain (any number of) digits are discarded, as well as tokens that appear less than 4 times in the entire dataset, or that appear in more than 0.1% of the documents in the dataset. This leaves us with a bag of tokens that can be expected to be rather specific to certain entities in the KB.

During the second parse, on a per article-, and thus per entity basis, the same normalisation is applied, and all tokens that appear both in the WikiText, and the bag of tokens generated during the first parse, are retained as being specific Context words for that particular KB entity.

3.1.4 WikiFacts

This is simply a dataset of selected WikiFacts, listed per entity. In practice, the "selected" may be interpreted as "almost all". Nevertheless, it is possible to exclude specific facts, like those that contain only numerical information, since we found that they have a negative influence on the performance of our system due to the fact that these same numbers often appear in articles unrelated to the entity.

We would also like to point out that, since the TAC KB has been parsed from a Wikipedia snapshot, it also suffers from the same problem of having no standard syntax for the factnames. As a consequence, many different variations exist for several facts. Consider e.g. "birthplace", "birth place", "birth_place", "Bplace", etc. To somewhat try to counter this, we created normalised versions of all the factnames, and treated all facts that generate the same normalised form as equal.

3.1.5 WikiLinks

WikiLinks is a dataset extracted from the KB, containing, for each entity (which is in essence a Wikipedia article), a list of all incoming and outgoing references to other KB entries. The incoming links per entity are referred to as the "WikiLinksIn", whilst the outgoing links are the "WikiLinksOut".

To make things more clear, take a look again at Figure 1. There you will notice that the fifth line, the `cityofbirth` fact, contains a tag `<link entity_id="E0467057">Manchester</link>`. This means that for the entity `E0000001` there is an outgoing link to entity `E0467057`, so "Manchester" is a WikiLinkOut for "Mike Quigley", while "Mike Quigley" is an "incoming link" for "Manchester". Note that for big cities like Manchester, New York, Paris,... the number of incoming links tends to be very large.

3.2 Processing the Articles

The TAC article dataset consists of literally millions of documents from different sources (newspapers, blogs,...), the vast majority of them contained in individual files. Since working with the individual files proved highly unpractical, we first grouped the articles in batches of approximately 10.000 articles each. Then, we applied NER on all article texts. From the output of the NER, we created two datasets per label (LOC, ORG, PER, MISC):

- A dataset containing a list of all entities of that label in the considered dataset, as tagged by the NER system, followed by the number of times it appears in the entire dataset, and the number of distinct articles it appears in; e.g. the PER variant of this dataset contains the line "`Barack Obama 48463 41024`", telling us that "Barack Obama" has been tagged as a PER 48463 times in the entire dataset (in casu, all documents from the 2009 TAC English Entity-Linking task) in 41024 distinct articles.
- A dataset containing a per article listing of all entities of that type, as well as the number of times they appear in that particular article (see Figure 5).

```
#doc AFP_ENG_19940517.0172.LDC2007T07
PERs:  3
Jamaluddin Omar  1
Olli Heinonen  1
Vladmir Rukhlo  1
```

Fig. 5: Extract from the PER dataset

4 Entity Linking

Starting from the datasets described in §3.2, we parse all documents separately, per label, except for MISC. There is a generic model for the three remaining labels, with some specific tweaks per label. Recall that the labels for the Stanford NER and the TAC KB differ. In this regard, we equate "LOC" to "GPE". When parsing a specific label, we treat all UKN's in the TAC KB as if they were of that specific label.

In a first step, we cluster the mentions of the label under consideration on a per document basis (see §4.1). Each cluster is identified by one of its members. We refer to this member as an "InnerID". Next, we try to resolve these InnerIDs to reference entities form the TAC KB, and if no suitable matches are found, we tag them as being a NIL, the identifier used by TAC to show that a mention is not present in the TAC KB.

The generic steps we go through when resolving these InnerIDs are as follows:

1. Per document, retrieve the list of InnerIDs of the label being parsed.
2. Per InnerID, gather a list of possible candidates from the TAC KB; if specific conditions are met, a specific candidate is immediatly returned as match.
3. If no candidates are found, normalise the InnerID, and try again.

4. If still no candidates are found, assume there are no matches in the KB, and tag the InnerID as a NIL.
5. Else, score all candidates (even if there is only one candidate!).
6. Return the candidate with the highest score *if* it exceeds a certain threshold, else tag the InnerID as a NIL.

In what follows, we will delve deeper into these different steps.

4.1 In-article Clustering

Before resolving mentions of a specific label over the entire dataset, they are clustered per article. Usually, the longest item per such cluster will become the item by which the cluster is identified, and we refer to it as an InnerID, and the cluster will contain all substrings of this InnerID that appear in the article. Suppose, e.g. an article contains the PERs "Obama", "Barack Obama", and "Mitt Romney", then the first two will define one cluster, while the last one will form a cluster on its own. The two InnerIDs of this article are then "Barack Obama" and "Mitt Romney".

In some cases, the InnerID will not be the longest cluster item, though. During clustering, we take into account all known entities from the KB. Suppose e.g. that the NER engine tagged "Barack Obama Administration" as a PER in some document, as well as "Barack Obama". Then our system will recognise "Barack Obama" as a reference entity, and hence make this the InnerID, instead of the longer "Barack Obama Administration". In other words, we give priority to names that are known from the TAC KB.

It is also possible to force an InnerID for some document, i.e. to force the system to add some desired string as an InnerID to some document. This is necessary for forcing TAC queries whenever the mention to be resolved has not been detected by the NER system, or when a mention has been detected, but tagged with the wrong label.

4.2 Finding Candidates

We will start by introducing some terminology. Recall that we store a list of surface forms for each KB entity (Fig. 3). When resolving a mention, we first look for candidates by performing string comparisons on this dataset of surface forms. We distinguish two ways in which a mention can match an entity:

- The mention is an exact match with the canonical name of an entity (or entities, if this name is ambiguous); in this case, we say the mention is an **ExactRefID**.
- The mention is an exact match with one of the surface forms of at least one entity; in this case, we say the mention is a **FuzzyMatch**.

Note that an ExactRefID is always a FuzzyMatch, but not the other way around. Also, we ignore the disambiguation part of a KB Entry when determining when a match is exact or not. In other words, going back to the example in Figure 1, even though the "exact" name would be "Mike Quigley (footballer)", we discard the "(footballer)" part. We do not discard the "after the comma" part though, meaning e.g. that "Houston" is a FuzzyMatch for "Houston, Texas".

Next, we will proceed by specifying how candidates are gathered for each label specifically.

4.2.1 LOC

For locations, first we will check whether we recognise the mention as an ExactRefID or FuzzyRefID. If so, we add all appropriate matches to the list of candidates. We also perform a check to see if the mention is a "KnownAlt", i.e. an abbreviation or alternative form of some location. Typically, these are abbreviations of American states. Consider e.g. "New York", which typically also appears as "NY", "N.Y." or "N. York". If a mention is recognised as an alternative form, and the full name(s) to which it refers are known as ExactRefIDs to us, we will add all FuzzyMatches for this full name to our list of candidates. Returning to our "New York" example, suppose the mention we are trying to resolve is "NY", then we will recognise this as an alternative form for "New York". There is more than one place with this name, however, and so by adding all FuzzyMatches, we include all these different New York's in our list of candidates.

4.2.2 ORG

Something that is typical for organisations, is the use of abbreviations. As a simple example, consider

"CIA". Almost nobody ever refers to this institution by its full name, "Central Intelligence Agency". Hence, when gathering candidates for ORGs, we first check whether the mention we are processing is likely to be an abbreviation, and if so, whether it is "explained", i.e. whether it is written out in full, in its vicinity.

A mention is considered likely to be an abbreviation if it contains no whitespaces, and is written in all uppercase. If this is indeed the case, we will search the article for a "written out" form by searching the article for a sequence of tokens whose first letters are uppercased, and when combined form our mention. In this process, we ignore the stopwords "of", "for", "the" and "and". In case we can actually find a written out form of the assumed abbreviation this way, which we call the "explanation", we will check if we recognise this explanation as a non-ambiguous reference organisation. If so, we resolve the mention to this reference entity. If not, we continue looking for candidates, but using the explanation as seed, rather than the original mention.

If we did not find an explanation, or if the mention did not appear to be an abbreviation in the first place, we will check if the mention ends on a known typical form for company types, like e.g. "gmbh", "corp.", "sprl", "ltd.", etc. If this is the case, we remove this part from the mention. E.g. "Coca-Cola Corp." will be modified to "Coca-Cola", and we will continue our gathering of candidates with this modified form.

First, we check whether the mention is a non-ambiguous ExactRefID. If this is the case, and as a fuzzy match it only points to one entity, resolve the mention to this entity. If it is an ambiguous ExactRefID, or a FuzzyMatch, add all fuzzy matches to the list of candidates. If there are no fuzzy matches, conclude that no suitable candidates could be found for this mention.

4.2.3 PER

First, it is important to note that, differing from GPEs and ORGs, the reference PERs are treated in a slightly different way, owing to the fact that a name can typically be split into a first name and a surname part (note however that his is not necessarily the case, as for example with artist names such as "Prince"). We use this to split every PER from the TAC KB whose canonical name contains at least one whitespace, at the occurence of the first whitespace, effectively dividing the name in a first and surname part. We use this to create an index over the entire PER collection of which KB entries contain which first- and surnames. This allows us later on to effectively retrieve all candidates for mentions that are only partial names. For this purpose, names that cannot be split are treated as first names.

Upon resolving a PER mention, we first apply normalisation rules to this mention: all lowercase, except for the first letter of each part of the name. E.g. "BOB doe" will become "Bob Doe". We refer to this process as "ForceCasing", and will refer to the forcecased mention as the MentionFC[3].In case the MentionFC contains at least one whitespace, it will also be split into a first- and surname at the occurence of the first whitespace. Hence our "Bob Doe" will be split into a firstname part "Bob", and a surname part "Doe".

If the MentionFC can indeed be split into two parts, we check whether the firstname is a known, popular nickname, using a list[4] of common nicknames.

- If this is not the case, we will check whether or not the MentionFC is a non-ambiguous ExactRefID, in which case we will resolve this mention to this entity. Else, we continue our gathering of candidates.
- If the first name is indeed recognised as being a nickname, we will substitute the nick with the name for which it is a short, and check whether this new form is recognised as a FuzzyMatch, and if so, will add the corresponding entities to our list of candidates. Going back to "Bob Doe", we recognise "Bob" as a popular nickname for "Robert", and will check whether or not "Robert Doe" is a FuzzyMatch, and if so, add all entities of which "Robert Doe" is a surface form to our list of candidates.

Note that "Bob" is a nickname for "Robert", but not the other way around, i.e. we will not replace "Robert" with "Bob".

Next, we check if the MentionFC is a "KnownName". Recall that we stored the PER database

[3] A few typical stopwords are ignored whose casing can make a difference.

[4] http://www.censusdiggins.com/nicknames.htm

into a matrix, tracking which first- and surnames are constituents of which fullnames, and thus, entities. Checking for a "KnownName" is simply verifying whether or not the MentionFC is recognised as being such a constituent (i.e. either a known first name, or a known surname). If this is indeed the case, we can efficiently retrieve all entities of whom this MentionFC is a constituent, and add these to our list of candidates.

Last, we will check if the MentionFC is a FuzzyMatch, and if so, will add all corresponding matches to our list of candidates.

4.2.4 Double Check

In case the above steps did not result in any suitable candidates, we will try the same steps again using a normalised form of the mention. This normalised form is simply a lowercased version of the mention, with all whitespaces and dots removed. Recall that for each KB entry, we generated a surface form in similar fashion. The idea behind this double check is to make the system more robust against small spelling mistakes, like different casing between mention and matching entity, or accidentally misplaced whitespaces, etc.

If this second parse still yields no suitable candidates, the mention is considered to be a NIL.

4.3 Scoring Candidates

In order to compute a global score for all candidates, six features are considered: WikiBankScore, WikiCtxtScore, WikiFactScore, WikiLinkInScore, WikiLinkOutScore, and a sixth feature which is the number of these previous features that is not zero.

WikiBankScore: this is an indication of the named entity overlap between the article in which the mention to be resolved appears, and the entities found in the WikiText of a candidate for that mention. Its value is computed as follows:

- for LOC and ORG and MISC, add 1 per entity that appears both in the article and the WikiText, equating LOC to GPE and MISC to UKN,
- for PER, add 1 per InnerID from the article that also appears in the WikiText,

For this purpose, we ignore the mention itself. Admittedly, the use of mapping MISC to UKN can be debated, as in essence, both are different creatures.

WikiCtxtScore: this is simply the number of tokens that constitute the WikiCtxt for the candidate under consideration that also appear in the article that is being processed. Note that during this scoring, the article does nog get normalised, because time-wise, this simply proved to be unfeasable.

WikiFactScore: number of WikiFact values that appear in the article. We discarded WikiFacts having all numerical values, typicially dates or coordinates or the like, because we noticed it would make for a lot of noise in the scoring, as numbers from these WikiFacts would fit unrelated, but equal, numbers appearing in articles. For similar reasons, wikifacts having generic values like "true/false", or "north/east/. . . " etc., were also discarded.

WikiLinksScore: this number indicates how many of the KB entities that are connected to the candidate also appear in the article. Specifically, for each candidate, per (lowercased) surface form of all WikiLinks for this candidate it will be checked whether or not it appears in the (lowercased) article or not. If it does, the WikiLinkScore gets incremented by 1.

For scoring purposes, as yet not difference is made between the incoming and outgoing links. Nevertheless, they have been split, so as to allow for possible experimentation with giving different weights to both.

4.3.1 Combining the WikiScores

After these individual WikiScores have been computed, we combine them into a proper scoring function. This function differs slightly for each label, giving more, or less, weight to specific WikiScores. These small differences have been tuned manually. In the following equations, $s\,(\text{WikiScores})$ represents the scoring function, its index indicates to which label it purports, and c represents the WikiCtxtScore.

$$\begin{aligned} s_{\text{GPE}} \;=\; & (\log(\text{e} + \text{WikiFact} + \text{WikiBank}) + \\ & \text{WikiLinkOut} + \text{WikiLinkIn}) * \\ & \left(1 + \sum_{i=1}^{c} \frac{1}{1+i}\right), \end{aligned} \tag{1}$$

$$s_{\text{ORG}} = (\log(\text{e} + \text{WikiFact}) + 2 * (\text{WikiLinkOut} + \text{WikiLinkIn}) + \text{WikiBank}) * \left(1 + \sum_{i=1}^{c} \frac{1}{1+i}\right), \quad (2)$$

$$s_{\text{PER}} = (\log(\text{e} + \text{WikiFact}) + \text{WikiLinkOut} + \text{WikiLinkIn} + \text{WikiBank}) * \left(1 + \sum_{i=1}^{c} \frac{1}{1+i}\right), \quad (3)$$

4.3.2 Afther the Scoring

Once a score has been computed for a candidate from the five WikiScores, it can then be multiplied with bonus or penalty factors. All labels share a same bonus and penalty factor. GPEs have a few additional factors. The penalty factor is the "UKN-Penalty", which is applied in case the candidate is a UKN (§4.3.3).

The bonus factor shared between all three labels is "DifferenceFound". Whilst scoring the candidates, the algorithm will check if there is a difference between the mention and the candidate, and whether or not this difference appears in the article. If it does, the score gets mutliplied by a factor 1.5. As an example, suppose you want to resolve the mention "Atlanta". Amongst the possible candidates for this mention will figure "Atlanta, Illinois", "Atlanta, Kansas", etc. In this case, the difference between "Atlanta" and "Atlanta, Illinois" would be "Illinois", and if the article in which the mention appears also contains "Illinois", the score for this candidate will be multiplied by 1.5.

For GPEs, in the same vein as "DifferenceFound", in case of candidates that contain an "after the comma" part, we compute a bonus factor as follows. Consider again the example of the mention Atlanta, which would have a.o. "Atlanta, Illinois" as candidate. Our system will check if this "after the comma" part, in this case "Illinois", is a FuzzyMatch. If so, we temporarily keep aside all fuzzy matches for "Illinois" that *do not* have an "after the comma" part (i.e. that are less likely to be ambiguous). For all of these fuzzy matches, we will then compute a score which is similar to Eq. 1. Finally, we take the average of all these scores over all these FuzzyMatches as a bonus factor for the original candidate. Supposing for simplicity's sake that we would have "Atlanta, Illinois" and "Atlanta, Kansas" as only candidates for Atlanta, and that the only FuzzyMatches to "Illinois" and "Kansas" would be themselves, we would then proceed to compute a score for these two cities in essence as if *they* were the mention to be resolved. This score for "Illinois" and "Kansas" would then count as a bonus-factor for "Atlanta, Illinois" and "Atlanta, Kansas" respectively.

Furthermore, also only for GPEs, if a candidate is an exact, case-unsensitive match to the mention, it also gets boosted by a factor 1.5. This is because locations are in general far more ambiguous than organisations or persons, and such boostings revealed necessary to push the correct candidate above all the noise.

Finally, if a candidate's final score exceeds a threshold that increases with the number of non-zero WikiScores (the sixth feature mentioned earlier), *and* it exceeds the highest score so far, then it will become the new best candidate. If after all candidates have been scored, none has been retained, the mention is considered to be a NIL.

4.3.3 Dealing with UKNs

If the candidate is an UKN, a penalty will be applied, since in essence you are not certain that this particular candidate is indeed of the same label as the entity you are resolving. The idea is to compute a score that indicates to what extend a KB UKN entry corresponds to one of the other labels. For this, a list of all (normalised) factnames was generated for the GPE, ORG and PER labels, as well as a list of all factnames that are *specific* to UKN.

In symbols: consider F to be the collection of all types of facts (i.e. normalised factnames) that appear in the KB. Define $F_{\text{GPE}} = \{\forall f \in F | \exists e \in \text{GPE} : f \in F(e)\}$, with $F(e)$ the collection of facts of e, and analogous for ORG and PER, and $F_{\text{UKN}} = \{\forall f \in F | \nexists e \in \text{GPE} \cup \text{ORG} \cup \text{PER} : f \in F(e)\}$.

Then, for each UKN, we counted how many of its facts overlap with those for the other labels. Dividing this number (3 numbers in total, one for each non-UKN label) by the number of facts for this particular UKN entity, gives a number between 0 and 1

that gives an indication of how much this particular UKN entity is likely to be a GPE, ORG or PER. In symbols, for an entity $u \in$ UKN, define

$$\text{UKNScore}_{\text{GPE}}(u) = \frac{F(u) \cap F_{\text{GPE}}}{\#F(u)}, \qquad (4)$$

and analogous for the other labels.

When scoring candidates, each UKN candidate c_{UKN} gets multiplied by a penalty factor p_{UKN} equal to

$$p_{\text{UKN}}(c_{\text{UKN}}) = \Big[\text{UKNScore}_{\text{Label}}(c_{\text{UKN}}) * (1 - \text{UKNScore}_{\text{UKN}}(c_{\text{UKN}}))\Big]^{1/4} \qquad (5)$$

where $_{\text{Label}}$ should be replaced by the label being parsed.

4.4 NIL

A "NIL" is a mention that could not be resolved to an entry in the TAC KB. Part of the entity-linking task was not only to detect when a mention was a NIL, but also to cluster these NILs.

Our approach to this problem was rather minimal. Whenever a mention was found to be a NIL, we mapped it onto a normalised surface form by making it all lowercase, and removing "'s" and certain punctuation marks when present. All NILs whose normalised forms are equal are clustered together.

Crude as this method may be, it strangely enough worked better than a more advanced method we tried out. In that method, in a first intance we would cluster the NILs using the same basic approach stated earlier, but we would also keep track of which documents the NILs would appear in. After all documents had been parsed, we would then try to detect ambiguous NILs within a same cluster by taking one NIL from this cluster, and comparing the bag of entities, as tagged by the NER engine, from the document it appeared in, with the bag of entities of all other documents within that same cluster, the aim being to cluster all NILs whose corresponding documents had at least one entity in common. The B^3 F1+ scores we obtained this way for the 2011 data were on average slightly below that of the more naive approach.

5 Evaluation

To benchmark our system, we mainly focused on the TAC 2011 English Entity-Linking (EL) task queries, but also ran our system on the 2010 queries, which did not have the extra NIL-clustering task attached to it. A breakdown of the total number of queries and queries per label for the 2010 till 2012 EL tasks can be found in Figure 6. Note that the label of a query is not known during evaluation, but is only revealed within the annotated set.

Year	Total	GPE	ORG	PER
2010	2250	749	750	751
2011	2250	750	750	750
2012	2226	602	706	918

Fig. 6: Breakdown of queries for the 2010, 2011 and 2012 English EL Tasks

An example of a 2011 query can be seen in Figure 7. Note that the 2012 queries sport added offset data for the mentions, i.e. they include the positions of the mentions to be resolved in the article under consideration. Each query is given a unique identifier, and consists of a document identifier and the mention figuring in this document to be resolved. The solution to the particular query in Figure 7 is "`EL_00001 NIL290 PER`". We again see the unique query identifier, followed by either the unique identifier of the KB entity to which it should be resolved, or as in this case, "NIL" to indicate that it does not appear in the KB, followed by a cluster number, and finally we find the label of the mention. Note that the clusternumber itself is not important, only the fact that queries belonging to a same cluster carry the same number.

```
<query id="EL_00001">
<name>A. Raghuveer</name>
<docid>AFP_ENG_20080725.0242.LDC2009T13
</docid>
</query>
```

Fig. 7: TAC 2011 EL query example

We used the information from the annotated queryset to split the queries into three subsets, grouping queries per label, so as to be able to focus on one label during development.

5.1 Parsing the queries

A difference should be made between "training" runs, in which we assume we already have the correct answers at our disposal, and "evaluation" runs, for which we only have the queries. The big difference between the two, as far as our system is concerned, is that for the training runs, we *know* what label the mentions carry.

5.1.1 Training Runs

When performing a training run, we let our system run as it would run on any other article, except for one thing. We known what mentions we should find in what documents, as well as what label these mentions carry. Hence, when letting our system parse a label, and we notice that we do not find a certain query that we should normally find, i.e. the mention we are supposed to resolve has not been found in the article in which it should be found, or it has been found but tagged with a wrong label, we force the query. This means that we forcefully add the mention to be resolved as an InnerID of its label to the document under consideration.

5.1.2 Evaluation Runs

When performing an evaluation run, we do *not* know the correct label of the mentions to be resolved. Taking this into account, and recalling that our system does not predict labels for mentions, we devised two types of evaluation runs.

Type 1: in this type, we first parse each query per label. Whenever the mention for a query has been found within the label elements which are being parsed, the query is parsed. If its mention has not been found, we skip it, but store it as being unprocessed for this particular label.

When all three labels have been parsed, we then take all queries that have not yet been parsed by any of the labels, and forcefully parse them for each label. That means that these queries get parsed thrice, by being forced once for each label. Afterwards, a series of rules, designed using the 2011 training data, is applied to select one of the three labels as being the most probable answer.

Type 2: in this type, we parse all queries thrice, forcing a query whenever its mention was not found in the corresponding document.

5.2 Results

The results of our test runs, as obtained using TAC's own "ELScorer" scoring tool, can be found in Figure 8. Our system performs reasonably consistent on the 2010 and 2011 data, but significantly less on the 2012 data. This shows the increased query difficulty. Furthermore, our system performs remarkedly better for PERs than for ORGs and GPEs. Although the difference in efficiency between ORG and PER is more or less stable across the datasets, GPEs take a noticeable extra hit for 2012, due to the extreme ambiguity of the queries. This makes that the vast majority of queries return a large list of candidates, with typically only little contextual information to disambiguate between them, resulting in many wrong resolutions and missed NILs. Also, the severe drop in B^3+ F1 score between 2012_Eval1 and 2012_Eval2 suggests that our system is very sensitive to wrong mention label predictions. Finally, the considerable departure from the average difference between μ-average and B^3+ F1 scores of about 3% for 2012Tr_ORG suggests a considerably higher number of ambiguous NIL queries for ORGs, something that our primitive NIL-clustering method is not armed for.

6 Conclusion & Future Work

We presented a rule-based entity-linking system that is label-specifically tuned through different candidate scoring, and the tackling of problems such as nicknames, American vs. British spelling, and abbreviations for organisations. Our system parses the data on a per-label basis. Overall performance is 58,6% B^3 F1, 5% above the median.

Future work includes the rigorous optimisation of the scoring functions, and adapting our system to parse all labels simultaneously. This would pave the way to updating our system to reason over all mentions and candidates at once, and thus take dependencies between candidates into account, rather than resolving each mention independently from all others.

Acknowledgement

This work was funded by iMinds (the former IBBT), Belgium.

Run	μ-avg.	B^3 P	B^3 R	B^3 F1	B^3+ P	B^3+ R	B^3+ F1
2010Eval_1	0.806	-	-	-	-	-	-
2010Eval_2	0.793	-	-	-	-	-	-
2010Train	0.832	-	-	-	-	-	-
2010Tr_GPE	0.752	-	-	-	-	-	-
2010Tr_ORG	0.804	-	-	-	-	-	-
2010Tr_PER	0.939	-	-	-	-	-	-
2011Eval_1	0.794	0.960	0.936	0.948	0.768	0.760	0.764
2011Eval_2	0.773	0.950	0.936	0.943	0.744	0.745	0.744
2011Train	0.828	0.964	0.942	0.953	0.802	0.795	0.799
2011Tr_GPE	0.780	0.971	0.914	0.942	0.765	0.737	0.751
2011Tr_ORG	0.796	0.943	0.953	0.948	0.752	0.771	0.762
2011Tr_PER	0.907	0.979	0.958	0.968	0.890	0.877	0.883
2012Eval_1	0.656	0.809	0.913	0.858	0.561	0.615	0.586
2012Eval_2	0.593	0.700	0.911	0.792	0.437	0.553	0.488
2012Train	0.739	0.843	0.923	0.881	0.646	0.696	0.670
2012Tr_GPE	0.613	0.830	0.945	0.884	0.548	0.596	0.571
2012Tr_ORG	0.720	0.783	0.829	0.805	0.556	0.614	0.584
2012Tr_PER	0.836	0.948	0.980	0.964	0.808	0.824	0.816

Fig. 8: Results for TAC 2010, 2011 & TAC 2012 query sets. "Eval_1/2" refers to evaluation runs of type 1 and 2, "Train/Tr" to training runs. For training runs, we have also included focussed results for each label.

References

A. Bagga and B. Baldwin. 1998. *Entity-Based Cross-Document Coreferencing Using the Vector Space Model*. Proceedings of the 17th international conference on Computational Linguistics, Vol. 1, pp. 79-85

W. M. Song, H. T. Ng and D. C. Y. Lim. 2001. *A Machine Learning Approach to Coreference Resolution of Noun Phrases*. Computation Linguistics, Vol. 27 Issue 4, pp. 521-544

C. H. Gooi and J. Allan. 2004. *Cross-Document Coreference on a Large Scale Corpus*. Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2004, pp. 9-16

J. R. Finkel, T. Grenager and C. Manning. 2005. *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling*. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370

R. Bunescu and M. Pasca. 2006. *Using Encyclopedic Knowledge for Named Entity Disambiguation*. Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, pp. 9-16

S. Cucerzan. 2007. *Large-Scale Named Entity Disambiguation Based on Wikipedia Data*. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 708-716

D. Milne and I. H. Witten. 2008. *An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links*. Proceedings of the Association for the Advancement of Artificial Intelligence

A. L. Gentile, Z. Zhang, L. Xia and J. Iria. 2009. *Graph-based Semantic Relatedness for Named Entity Disambiguation*. Proceedings of the 1st International Conference on Software, Services and Semantic Technologies

X. Han and J. Zhao. 2009. *Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge*. Proceedings of the 18th ACM conference on Information and knowledge management, pp. 215-224

J. Hoffart, M. A. Yosef, I. Bordino, H. Frstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater and G. Weikum. 2011. *Robust Disambiguation of Named Entities in Text*. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 782-792

A. Pilz and G. Paass. 2011. *From Names to Entities using Thematic Context Distance*. Proceedings of the 20th ACM international conference on Information and knowledge management, pp. 857-866