UGent Participation in the TAC 2013 Entity-Linking Task

Laurent Mertens, Thomas Demeester, Johannes Deleu, Chris Develder
Dept. of Information Technology - IBCN
Ghent University – iMinds, Ghent, Belgium
firstname.lastname@intec.ugent.be

Abstract

This article describes the system used by the UGent-IBCN team for participating in the Text Analysis Conference (TAC) 2013 English Entity-Linking task. We kept the overall rule-based workflow of our last year's submission, but significantly altered individual components. Most importantly, these changes include improved document pre-processing, new ways of candidate selection, and completely redesigned scoring and NIL-detection mechanisms. Finally, we provide detailed data of our system's performance.

1 Introduction

The aim of the TAC Entity-Linking (EL) task is to build a system that resolves given mentions in a given set of documents to their corresponding entities in a standard Knowledge Base (KB), referred to henceforth as the TAC KB. Systems should also be able to detect the case in which the underlying entity of such mentions is not represented in the TAC KB, and cluster these mentions (i.e., surface forms) when they refer to the same entity. KB entities are assigned one of four possible labels: GPE (Geo-Political Entities), ORG(anizations), PER(sons) or UKN (unknown). This last label should be considered a wildcard signifying the entity could be anything, rather than indicating the entity is neither of the other labels.

The system we used for participating in this year's task is a direct evolution of last year's system, which is described in detail in (Mertens et al., 2012). The overall workflow has not been altered, but the separate components have undergone significant changes. As such, we will often refer to our last year's paper for additional details where relevant, in order to reduce redundancy, and focus on the novelties.

In what follows, we will start with a general overview of our system, and zoom in more closely on the pre-processing of the data, highlighting differences with last year's system in Section 2. Section 3 will focus on the entity-linking proper, followed by Section 4 in which we provide ample data demonstrating the accuracy of our system on the TAC 2010, 2011, 2012 and 2013 English EL tasks. We conclude with some final remarks, and possible roads for further work in Section 5.

2 Data Preparation

2.1 General Overview

The general workflow of our system can be divided into two distinct steps. First, there is the preprocessing of the KB and documents to be parsed, followed by the combination of all these elements in order to parse the documents proper, and thus resolve the mentions they contain. An overview of the workflow of our system is depicted in Figure 1.

The rationale of the pre-processing is to group different facets of the information contained in the KB into distinct repositories, to ease manipulation of this data and keep things easier to oversee. The document representation (after preprocessing) should allow a practical comparison between document contents and the relevant KB entries.

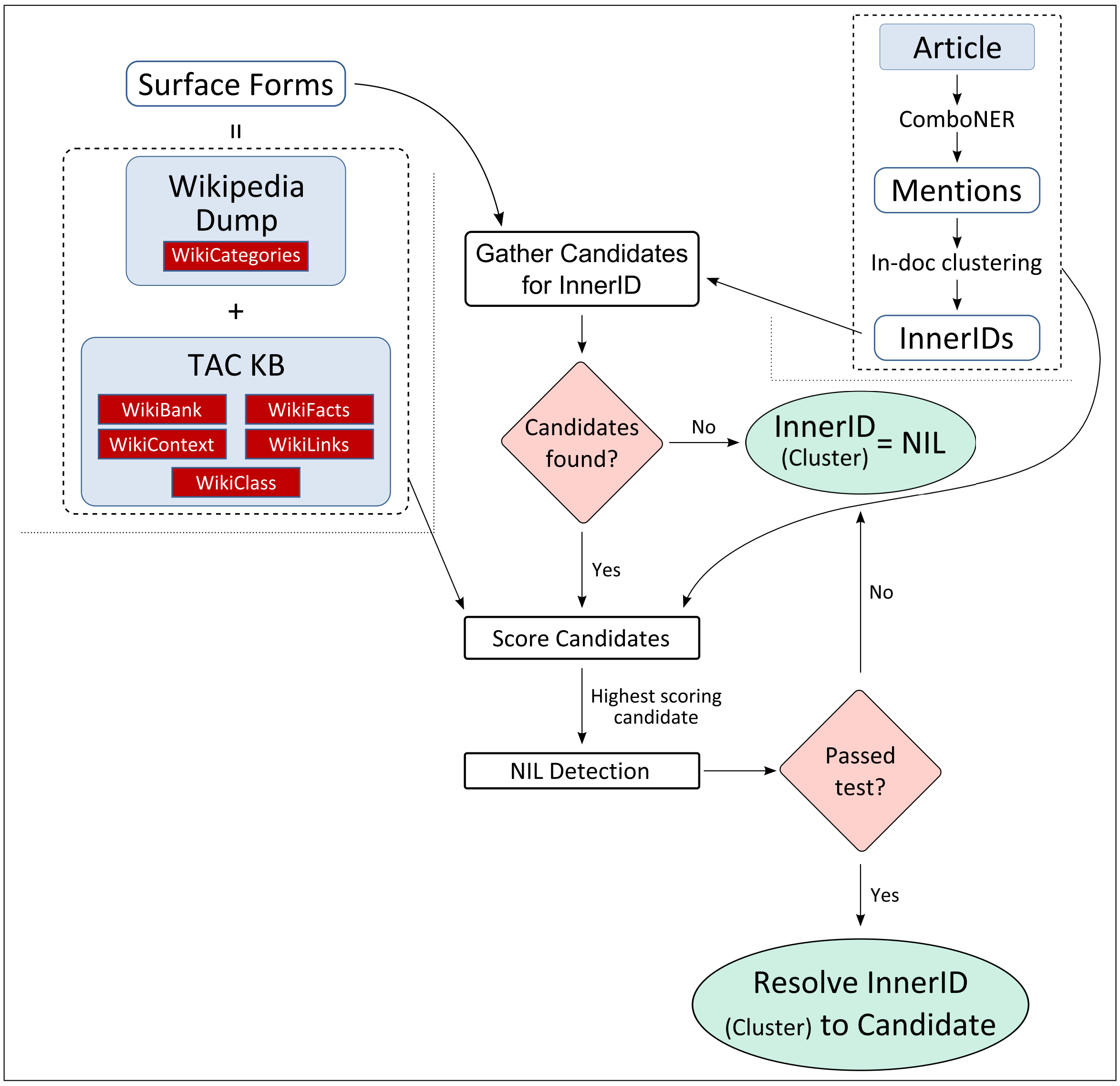

Fig. 1: Flowchart

2.2 Combining NER Systems

A key ingredient in this pre-processing is applying Named Entity Recognition (NER) to the relevant texts. For this, we combined four different NER systems, namely CogComp (Ratinov & Roth, 2009), LingPipe[1], OpenNLP[2] and Stanford NER (Finkel et al., 2005), as discussed below.

To annotate a document D, we first apply the different NER systems to it. The Stanford NER is used twice, using a case-sensitive and case-insensitive model. In total, this amounts to five different NER outputs, resulting in five tokenized and annotated documents. To allow a token-per-token comparison between systems, we need to first align these outputs, as different NER systems use different tokenization rules.

After alignment, we proceed to the mapping of the annotations to one “combined” annotation, using performance statistics of each NER obtained using the annotated TAC queries from previous years. Finally, we post-process the results to further reduce the number of wrong tags.

We refer to this system as the ComboNER. Performance statistics for the TAC 2012 and 2013 query

[1]http://alias-i.com/lingpipe/index.html
[2]http://opennlp.apache.org/

		BIS	Avg.	Combo
	Total 2013 queries: 2190			
2013	Correct	1368	1142	1620
	Exact	1331	1075	1500
	Fuzzy	37	67	120
	Total 2012 queries: 2226			
2012	Correct	1673	1417	1865
	Exact	1636	1369	1796
	Fuzzy	37	48	69

Table 1: NER statistics on TAC 2012 and 2013 query sets. "BIS" = Best Individual System, "Avg." = average over individual systems.

sets is shown in Table 1. In this table, "Correct" means the NER annotation between the query offsets in the query document bears the query label, "Exact" means the query offsets correspond to the mention boundaries, and "Fuzzy" means the mention between the query offsets is a subset of a larger mention. An example of this last instance would be the case were the query mention is e.g. "Obama" at query offsets [start = 255, end = 259], but the fully annotated mention is "Barack Obama" at position [start = 248, end = 259].

2.3 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 2.

Each entry starts with a line containing the title of the Wikipedia page from which the information was 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 class assigned to the original infobox (i.e., one class per entity), and 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". Recall the possible label values: GPE, ORG, PER and UKN. Note the difference with the "typical" NER labels: LOC(ation), MISC(ellaneous), ORG and PER.

2.3.1 Splitting the KB

We attempt to dissect the KB into several distinct parts, defined by the type of data they contain. To this end, we define and create directly from the TAC KB:

- **WikiBank:** a dataset containing all named entities from the WikiTexts, as tagged by the NER system, per KB entry.
- **WikiClass:** a dataset containing the corresponding infobox class (see above) for each KB entry.
- **WikiCtxt:** a dataset containing relevant contextual terms for each KB entry.
- **WikiFacts:** a dataset containing all (factkey, factvalue)-pairs for each KB entry, extracted from the TAC KB.
- **WikiLinks:** a dataset containing for each KB entity a list of all incoming and outgoing references to other KB entries.

For details on how these datasets were created, we refer the reader to (Mertens et al., 2012), as this part of the process has remained practically unaltered. The WikiClass is new, but is in essence merely a mapping between entities and infobox classes. Significant differences occur for the WikiBank, and WikiCtxt. For the WikiBank, we created a new WikiBank by applying the ComboNER to the WikiTexts, instead of only the Stanford NER. For the WikiCtxt, we altered the system to work with stems, rather than tokens. For this, we used the English stemmer provided with the Lucene[3] package.

Also a new addition to the troops is the **WikiCats** dataset, which we created from a June 2012 Wikipedia dump, and which contains all Wikipedia categories relevant to each KB entry. For this, we first gathered a list of redirects from TAC KB entities to new Wikipedia pages, and then parsed the Wikipedia pages that correspond to an entity in the TAC KB, either directly or through a redirect, and extracted its corresponding Wikipedia categories.

[3]https://lucene.apache.org/

```
<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. 2: Excerpt from the TAC KB

2.3.2 Cleaning up UKNs

The majority of the entities in the KB are UKNs. This makes that when parsing, one has to treat them as potentially of any label, which for our system results in a lot of noise when generating candidates. In an attempt to alleviate this problem, we also created a list containing, for each label, all WikiClasses that appear with entities of this label (see Figure 2), as well as how many times each class occurs for each label. For this purpose, we normalized the class names[4]. This list is stored in a simple tab-delimited text file, and by adding a tag preceding a class, we can "route" this class to the label specified by the tag, or ignore all entities bearing this label for classes tagged with "@REM". As this task needs to be done manually, and there are a lot of different UKN classes (2238 normalized classes, to be precise), we did not annotate all classes, but only the most commonly occurring. A short excerpt can be seen in Figure 3 to illustrate this. Statistics on how many classes and entities were redirected for our submission runs can be seen in Table 2.

```
#GPE
settlement    88301
uk place      9102
...
tw district   1
#ORG
radio station 7333
military unit 5945
...
#UKN
@REM          album            72992
@REM          film             34659
@PER          musical artist   30092
...
```

Fig. 3: Excerpt from the class redirect list.

2.3.3 Extracting Surface Forms from the KB

This part has essentially remained unchanged. See (Mertens et al., 2012) for details.

		GPE	ORG	PER	REM
ORG	C.	3	0	0	1
	E.	1007	0	0	31
UKN	C.	85	64	113	212
	E.	113432	40357	79304	262666

Table 2: Class (C.) and Entity (E.) label redirect statistics. Unlisted combinations indicate 0 redirects.

[4]The normalizing procedure consisted of lowercasing all names, removing the "infobox" part, replacing underscores by whitespaces, and converting `&amp;`, `&quot;`, `&lt;` and `&gt;` to the characters they represent.

2.4 Processing the Documents

Documents to be parsed are grouped into batches, simply meaning that we concatenate multiple documents into one single, large, document. The size of these batches is arbitrary, the limitation being determined by the amount of RAM of the system, as these batches get fully loaded into memory upon parsing. Each batch is comprised of 5 files. One file contains the tokenized and annotated (concatenated) document, as processed by the ComboNER. The four remaining files, one per label (LOC, ORG, PER, MISC), contain lists of mentions per document for the label under consideration, appearance counts of these mentions per document, as well as the positions at which these mentions occur. (Since TAC queries contain character offsets in the original documents, we had to translate token positions in our processed documents to these original character positions.)

The TAC 2013 EL corpus contains documents from three different sources, namely newswire data (NW), web data (WB) and discussion forum data (DF). Note that the DF data is a new source introduced this year.

3 Entity Linking

Starting from the datasets described in §2.4, we parse all documents separately, per label, except for MISC[5]. There is a generic model for the three remaining labels, with some specific tweaks per label. Recall that the labels used by TAC differ from the "standard" NER labels. In this regard, we equate "LOC" to "GPE". When parsing a specific label, we treat all UKN's in the TAC KB that remain after the mapping described in §2.3.2 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 §3.1). Each cluster is identified by one of its members, which we refer to as an "InnerID". Next, we try to resolve these InnerIDs to reference entities from the TAC KB, considering them "NIL" if no suitable matches are found. The linking result for the InnerID is then transferred to all members of its cluster.

[5]We do not parse for MISC, since there are no MISC queries.

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.
3. If no candidates are found, normalize the InnerID (i.e., its surface form), 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!).
6. Return the candidate with the highest score *if* it passes the NIL test, *else* tag the InnerID as a NIL.

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

3.1 In-document Clustering

This part has remained largely unchanged compared to our TAC 2012 system, except for some bug fixes. Note that it is also possible to forcefully 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 ComboNER system, or when a mention has been detected, but tagged with the wrong label. This last situation is only relevant to training runs (see §4.1).

3.2 Finding Candidates

We distinguish three ways to find valid candidates for a given mention. The first way is by string comparisons and query expansions, and has been explained in detail in (Mertens et al., 2012). It has since remained largely unchanged. The two remaining, newly added, methods, aim to find candidates in a more indirect way, as described next. Not all approaches are used for all labels; which approach is used for which label is displayed in Table 3.

3.2.1 Using Categories

One new method tries to exploit the WikiCategories. We first generate a list of categories relevant to the document, by, for each token from the document, retrieving the list of categories to which this token belongs (quite possibly none). A token is considered to belong to a category simply if it is contained in the category name. Stopwords are ignored in this regard. We then cross-check these lists to determine how many tokens of a given category appear in the document. If this number exceeds a given threshold[6], we retain this category. Then, we retrieve a list of all entities that have this category assigned to them. Finally, we perform some string comparisons in order to measure whether there is sufficient overlap between the candidate's name, and the mention under consideration. If so, the candidate is retained.

3.2.2 Using Context

A second way is by trying to exploit the WikiContext. The process is very similar. We stem the document, and for each stem retrieve a list of entities whose WikiContext contains this stem. We compute a score for each entity that takes into account the weight of the stem for this entity[7], as well as the total number of known context stems for this entity. If this score exceeds a given threshold, the candidate is retained.

	GPE	ORG	PER
Text	x	x	x
Cat	o	x	x
Ctxt	x	x	o

Table 3: Breakdown of candidate gathering approaches per label. (Text = string comparison & query expansion method.)

3.3 Scoring Candidates

The scoring of candidates is a three step process. First, we compute a number of subscores, focusing on different aspects of the KB. Then, we combine these scores into one final score by linearly combining them using optimized weights. Lastly, the candidate with the highest score is subjected to a NIL test. The overall scheme is the same for each label, but the weights are label-dependent, and also dependent on the number of candidates found for a given mention. We come back to this after first delving a bit deeper into these steps.

[6]More precisely, if the ratio of the number of category name tokens present in the document to the total number of tokens the category name consists of exceeds a given threshold.

[7]This weight is a function of the ratio of the number of times this stem appears in this entity's WikiText to the total number of occurences over the entire TAC KB.

3.3.1 Subscores

We compute the following subscores per candidate. For more details, see (Mertens et al., 2012).

WikiBankScore: this is a measure of the named entity overlap between the document, and the candidate's entry in the WikiBank.

WikiCatsScore: this is a measure of the overlap between the Wikipedia categories assigned to the document, and those associated with the candidate.

FullCatsScore: contrary to the WikiCatsScore which considers partial overlap between document and category name, the FullCatsScore counts only those categories whose every token appears in the document.

WikiCtxtScore: this is the number of stems the document and the candidate's WikiContext have in common.

WikiFactScore: this is the number of WikiFact values that appear in the document. Note that some WikiFacts, e.g., those containing only numbers, are discarded.

WikiLinksScore: this number indicates how many of the KB entities that are connected to the candidate also appear in the document. This score is actually split in two parts, the self-explanatory **WikiLinksInScore** and **WikiLinksOutScore**.

Non-Zero Feats: this is the number of WikiScores (see previous paragraphs) that are non-zero.

Exact CU Match: this is a boolean value indicating whether or not the InnerID is an exact case-insensitive match with the candidate's generic name (i.e., the one extracted from its Wikipedia page title).

Difference Found: this is a boolean value indicating whether or not the difference between the InnerID's and the candidate's names has been found in the document. E.g., consider the InnerID "Atlanta", and the candidate "Atlanta, Illinois". In this case, the difference between "Atlanta" and "Atlanta, Illinois" would be "Illinois". If this term appears in the document, this score will be 1, else it will be 0. Note that this check is actually case-insensitive, and that if the difference consists of more than one token, all tokens need to be present in the document.

3.3.2 Combining the Subscores

The combination of the subscores is done through a straight-forward weighted sum, with two twists.

Firstly, the WikiCats, WikiFacts and both WikiLinks scores are amortized, by taking into account how many items the candidate's entry in the respective datasets contains. This is done in order to account for the fact that one expects it to be more likely for overlap to occur for a candidate having, e.g., many WikiLinks, than for a candidate having very few WikiLinks.

Second, the weights are not only label-dependent, but also depend on the number of candidates that are found for the InnerID under consideration. Per label, we define five ranges, namely $[1, 2]$, $[3, 5]$, $[6, 10]$, $[11, 25]$, and $[26, +\infty[$. For each of these ranges, we determine a set of optimized weights.

After the combined score has been determined, we apply two more corrections as follows:

$$S = S_{\text{comb}} \cdot \text{UKNPenalty} \cdot \text{OW}. \qquad (1)$$

In this equation, S is the final score we obtain for a particular candidate, S_{comb} refers to the combined subscores, UKNPenalty refers to the factor described in §3.3.3, and OW refers to the Origin Weight, as described in §3.3.4.

3.3.3 Dealing with remaining UKNs

For UKN candidates that were not filtered by our WikiClassMap, we still aplied the same UKNPenalty from last year. For details, see (Mertens et al., 2012). In short, the UKNPenalty is a weight between 0 and 1 that indicates how much of a given non-UKN label (GPE, ORG, PER) the UKN under consideration appears to be. For non-UKN candidates, the UKNPenalty is simply 1.

3.3.4 Origin Weights

Recall from §3.2 and Table 3 that we use different approaches to generating candidates. Noting that one particular candidate can be gathered by more than one method, in total these three methods can be combined into seven different "origins" for each candidate. E.g., a candidate can be found by the string comparison method, as well as through use of categories, whilst another candidate was only found through use of context. For each of these origins, we assign a weight, the OW factor in Eq. 1, allowing to diminish or increase the role of a certain origin. E.g., one might expect that a candidate that has been found by all three methods is more likely to be correct than one that has been found by only one method. In practice however, rather the opposite appeared true, namely that some methods were noisier than others, and that this noise level was label-dependent.

3.4 Dealing with NILs

The final step in the InnerID resolution is to determine whether the highest scoring candidate is actually correct, or if this InnerID should rather be assigned the NIL tag. Additionally, NILs should be clustered according to their underlying entity.

3.4.1 NIL Detection

To determine whether or not to accept the highest scoring candidate, we use a NIL detection scheme comprised of six distinct NIL thresholds, each adding their fixed part to an overall NIL score if passed. If this score exceeds a final, seventh, threshold, the candidate is accepted. If not, we reject it, and thus assign a NIL tag to the InnerID. The individual thresholds are as follows.

- **NILNonZeroFeats:** a threshold comparing the top score to a function of the candidate's NonZero Feats (see §3.3.1).

- **NILRatio:** a threshold comparing the top score to a function of the ratio of the document's number of tokens to the document's number of mentions (regardless of label).

- **NILTopToSnd:** a threshold comparing the ratio of the top candidate's score to the second highest score to a fixed threshold. If there is no second candidate, this ratio is treated as $+\infty$.

- **NILTopToMean:** a threshold comparing the ratio of the top candidate's score to the mean score over all candidates to a fixed threshold. If there are less than 3 candidates, this ratio is treated as $+\infty$.

- **NILTopToTrd:** a threshold comparing the ratio of the top candidate's score to the third highest score to a fixed threshold. If there is no third candidate, this ratio is treated as $+\infty$.
- **NILWikiLinks:** a threshold comparing the top score to a function of the candidate's number of WikiLinks scores.

Note that NILTopToSnd and NILTopToMean actually use two thresholds each. If the highest of both thresholds is exceeded, this threshold will contribute more to the total NIL score than if only the lower threshold is exceeded.

Finally, we sum all individual threshold scores, and compare this to a final threshold. If this threshold is breached, the candidate is accepted, or else rejected.

These thresholds are a function of a number of parameters. These parameters are, just as for the scoring system, label-dependent, and also use the same 5 candidate ranges.

3.4.2 NIL Clustering

NIL clustering was done by mapping NIL InnerIDs to normalized forms, and clustering all InnerIDs with equal normalized forms together.

3.5 Parameter Tuning

The total number of parameters for our system is as follows.

- Per label: 7 origin weights. Note that not all weights are relevant to all labels, as not all labels consider all origins.
- Per label, and per candidate range:
  - 10 weights for the candidate scoring,
  - 16 weights for the NIL detection.

Optimization is divided into two distinct steps. First, we focus on the non-NIL queries to tune the scoring weights in order to maximize the number of queries for which the correct candidate obtains the highest score. Then, we turn to the full query set in order to tune the NIL detection weights, aiming for a maximal ("ordinary") F1 score[8].

[8]To be precise, we optimize the product of the F1 score with the number of correctly answered queries.

The basic scheme we follow in performing this optimization is as follows. For a given objective function $F(\vec{P})$ we wish to maximize, we (heuristically) define an initial parameter set $\vec{P_0}$, and a vector of step-sizes $\vec{\Delta}$, containing a default step-size for each parameter. Starting from $\vec{P_0}$, we then cycle through all parameter dimensions, for each parameter taking the corresponding step first in the positive direction, and, if this yields no improvement, in the negative direction. If a direction yields improvement, we keep stepping further down this direction until no further improvement is achieved, at which point we move on to the next parameter. Optimization stops when a full cycle without improvement has been gone through. One can then take the result of this optimization as a starting point of a next run using a different $\vec{\Delta}$.

4 Results

We trained our system using the TAC EL queries from 2010, 2011 and 2012. A breakdown of queries per label and year can be seen in Table 4. A breakdown of queries per source and year is shown in Table 5. Table 6 depicts statistics about the number of clusters, and number of entities per cluster, per year. Note that the clustering subtask was introduced in 2011.

Year	GPE	ORG	PER	Σ
2013	803	701	686	2190
2012	602	706	918	2226
2011	750	750	750	2250
2010	749	750	751	2250
Year	NW	WB	DF	Σ
2013	1134	343	713	2190

Table 4: Breakdown of queries per label and year, as well as per source for 2013.

4.1 Parsing the queries

We make a distinction between "training" runs, and "evaluation" runs. Training runs are runs for which we make use of the correct label assignment as found in the annotated queries. Evaluation runs do not make use of this information, and thus simulate a situation in which no annotations are available.

For training runs, we group the queries per label,

Year	NW	WB	DF	Σ
2013	924	289	607	1820
2012	1417	599	-	2016
2011	1486	745	-	2231
2010	1493	738	-	2231

Table 5: Breakdown of number of documents per source.

		All	Link	NIL
2013	#	726	327	399
	Avg.	3.02	3.33	2.76
	Med.	2	3	2
2012	#	1941	1016	925
	Avg.	1.15	1.16	1.13
	Med.	1	1	1
2011	#	1514	706	808
	Avg.	1.49	1.59	1.39
	Med.	1	1	1

Table 6: Cluster statistics. "#" = number of clusters, "Avg." and "Med." = average and median number of entries per cluster respectively.

and then parse each group separately. We parse the document for which the query is formulated, and if we do not find the query mention with the correct label in this document, we forcefully add it as an InnerID of the correct label.

For evaluation runs, we use the provided offsets whenever possible. These runs proceed in three steps when offsets are available, or two steps if not. In this last case, the first step is skipped.

Step 1: for each label, we go through all queries. If for a given label and query the query mention is found as being tagged with this label at the given position, then we go ahead parsing the query for this label. If not, we skip the query.

Step 2: for each label, we go through all queries that have not been parsed in Step 1. Offsets are not taken into account in this run, but we simply look if the given query mention has been tagged with this particular label in this document. If so, we parse this query for this label. Note that in this step, contrary to Step 1, a same query might get parsed for more than one label. In this case, heuristics are applied to choose between the different answers.

Step 3: all remaining non-parsed queries are queries for which the query mention has not been tagged as neither GPE/LOC, ORG or PER. We parse these queries thrice, once for each label, by forcing the mention as the label under consideration. Here again, heuristics are applied to choose between the different answers.

4.2 Statistics

We submitted several runs, but only list our best performing run in the following tables[9]. Tables 7 - 11 show different aspects of the performance of our system. Table 7 shows the results of our system on evaluation and training runs performed on the TAC 2010 till 2012 data, followed by Table 8 depicting detailed statistics of system performance on the TAC 2013 data. Table 9 shows candidate origin statistics for the combined 2010, 2011 and 2012 queries. Table 11 shows detailed information about the number of queries for which our system finds candidates, how many of those queries are actually "inKB" queries, for how many of these "inKB" queries we do indeed find the correct answer as a candidate, referred to as the "True Candidate", and in how many cases this True Candidate received the highest, second highest or third highest score (of all candidates for that query). The first two columns in this table denote the number of queries whose answer is "inKB" and NIL respectively.

4.3 Discussion

Although our system has vastly improved over last year's performance, as evidenced by the data depicted in Table 12[10], with evaluation runs significantly outperforming last year's training runs, and even outperforming last year's best system (0.744 vs 0.730 B^3+F1), it only performs slightly above the median in this year's task (0.600 vs 0.588 B^3+F1). We did expect sub-optimal performance on the DF documents due to their excessive length (see Table 10), something our system is not designed to

[9]The difference between these runs was chiefly the use of different parameter tunings.

[10]Note that some external components, e.g., the WikiBank, WikiCtxt, etc., have been updated since last year, as described in this paper, making the performance of our 2012 system on the 2013 data slightly better than it would have been with last year's KB components. The other results have been taken from (Mertens et al., 2012).

2010	μ-avg.	B^3+ F1
Eval	0.838	-
Ev_GPE	0.809	-
Ev_ORG	0.787	-
Ev_PER	0.919	-
Train	0.854	-
Tr_GPE	0.845	-
Tr_ORG	0.795	-
Tr_PER	0.923	-

2011	μ-avg.	B^3+ F1
Eval	0.855	0.823
Ev_GPE	0.824	0.798
Ev_ORG	0.805	0.761
Ev_PER	0.936	0.911
Train	0.870	0.840
Tr_GPE	0.849	0.826
Tr_ORG	0.819	0.776
Tr_PER	0.941	0.917

2012	μ-avg.	B^3+ F1
Eval	0.790	0.744
Ev_GPE	0.673	0.624
Ev_ORG	0.741	0.661
Ev_PER	0.905	0.885
Train	0.869	0.825
Tr_GPE	0.774	0.736
Tr_ORG	0.865	0.776
Tr_PER	0.935	0.918

Table 7: Results for TAC 2010, 2011 & 2012 query sets. “Eval/Ev” refers to evaluation runs, “Train/Tr” to training runs.

	2013	Eval	Ev_GPE	Ev_ORG	Ev_PER	Train	Tr_GPE	Tr_ORG	Tr_PER
All	μ-avg.	0.759	0.679	0.812	0.799	0.794	0.732	0.850	0.809
	B^3+F1	0.600	0.546	0.605	0.660	0.642	0.617	0.637	0.678
DF	μ-avg.	0.637	0.623	0.659	0.649	0.703	0.694	0.786	0.668
	B^3+F1	0.500	0.535	0.397	0.488	0.565	0.616	0.471	0.506
NW	μ-avg.	0.832	0.730	0.878	0.908	0.854	0.766	0.897	0.914
	B^3+F1	0.721	0.631	0.739	0.810	0.754	0.684	0.764	0.828
WB	μ-avg.	0.770	0.667	0.787	0.746	0.787	0.833	0.807	0.754
	B^3+F1	0.597	0.444	0.571	0.637	0.616	0.625	0.591	0.650

Table 8: Results for TAC 2013 query set. “Eval/Ev” refers to evaluation runs, “Train/Tr” to training runs. “DF” = Discussion Forum, “NW” = Newswire, “WB” = Web.

		I	Ca	I+Ca	Cx	I+Cx	Ca+Cx	All
GPE	TP	664	0	0	0	631	0	0
	FP	132	0	0	0	61	0	0
	NIL	78	0	0	6	28	0	0
ORG	TP	143	9	52	97	229	13	82
	FP	28	7	4	55	22	3	5
	NIL	25	4	2	32	20	2	0
PER	TP	545	2	173	0	0	0	0
	FP	45	0	3	0	0	0	0
	NIL	39	1	1	0	0	0	0

Table 9: Candidate Origin for training run on combined TAC 2010/2011/2012 queries. “I” = string matching, “Ca” = Category, “Cx” = Context. The NIL row contains the origin of the entities to which NIL queries were resolved.

	2013			2012		2011		2010	
	NW	WB	DF	NW	WB	NW	WB	NW	WB
Avg.	328	1959	13650	516	557	747	850	722	969
Med.	196	620	5316	344	380	610	705	593	553
Min	7	16	84	18	35	34	38	37	55
Max	3787	52997	134317	12398	4599	7102	3550	5201	16265

Table 10: Tokens per document statistics per document source and year, as extracted from the ComboNER output. “Med.” = Median.

		Queries		Candidates					
		inKB	NIL	QWC	Links	TCP	TC1	TC2	TC3
2012	GPE	527	75	525	523	502	437	31	11
	ORG	276	430	267	239	217	215	2	0
	PER	374	544	373	355	342	342	0	0
2011	GPE	229	521	518	511	498	480	9	6
	ORG	412	338	338	286	267	253	8	2
	PER	485	265	265	239	235	234	0	1
2010	GPE	503	246	500	492	470	452	12	1
	ORG	304	446	304	257	235	227	4	4
	PER	213	538	213	186	184	182	0	1

Table 11: Candidate Statistics for training runs. QWC = Queries With Candidates, Links = non-NIL queries, TCP = True Candidate is Present, TC# = True Candidate's score ranks at #.

data $\rightarrow$		2010		2011		2012		2013	
$\downarrow$ system		μ-avg.	B^3+F1	μ-avg.	B^3+F1	μ-avg.	B^3+F1	μ-avg.	B^3+F1
2012	Eval	0.806	-	0.794	0.764	0.656	0.586	0.264	0.137
	Train	0.832	-	0.828	0.799	0.739	0.670	0.368	0.188
2013	Eval	0.838	-	0.855	0.823	0.790	0.744	0.759	0.600
	Train	0.854	-	0.870	0.840	0.869	0.825	0.794	0.642

Table 12: System performance comparison between 2012 and 2013 submissions.

handle, but were initially disappointed by our performance on the other document sets.

When more closely analyzing the 2013 query set however, it is clear that much focus was directed toward the clustering subtask, as evidenced by the statistics shown in Table 6. Since we did not alter our primitive NIL clustering mechanism from last year, this explains in large part the massive difference of 15.9% between μ-average and B^3+F1 scores (2013 Eval, Table 8). Furthermore, when comparing μ-average scores on the NW and WB documents between the 2012 and 2013 query sets, as shown in Table 13, we note that our system performs as expected on this year's NW documents, and even scores better than expected on this year's WB documents, at least for the evaluation runs. Limiting ourselves to NW and WB documents for 2013, we obtain a weighted μ-average score of 0.806 for our evaluation run, which is in accordance with our μ-average score of 0.790 when evaluating the 2012 data. Recall that the 2012 queries were noticeably more difficult than previous years.

A noticeable difference concerning our system's performance on the 2012 and 2013 data, is the much smaller difference between evaluation and training run scores for 2013 (see, e.g., Tables 12 and 13). The clearly smaller selected NW documents for 2013 compared to all previous years (see Table 10) might provide a clue as to why this is for this type of documents, as the smaller size might limit the amount of NER errors, but we can only speculate in this regard, as this reasoning would, on first sight, contradict the same effect for the considerably longer WB and DF documents. Whatever the case may be, the decrease of the training-evaluation gap across the board when comparing our 2012 and 2013 systems indicates that our ComboNER system does indeed provide cleaner and more accurate NER tags, but this same gap still indicates that the dependency of our system on the NER accuracy is still its biggest weakness.

	NW		WB	
	Eval	Train	Eval	Train
2012	0.829	0.897	0.715	0.816
2013	0.832	0.854	0.770	0.787

Table 13: μ-average scores on NW and WB documents from TAC 2012 and 2013 query sets for 2013 system.

5 Conclusion

We explained the workings of our system used for participating in the TAC 2013 English EL task, focusing on changes and novelties compared to our 2012 system. Detailed performance statistics are reported concerning the performance of our 2013 system on several years of TAC EL tasks, as well as comparing the performance of our 2012 and 2013 systems. These last data show that our 2013 system has vastly improved over our 2012 system, scoring 15.8% and 46.3% higher on evaluation runs on 2012 and 2013 data respectively (0.744 vs 0.586 B^3+F1 for 2012, 0.600 vs 0.137 B^3+F1 for 2013), and beating last year's best system (0.744 vs 0.730 B^3+F1) on the 2012 EL task. Despite all this, our system performed only slightly above the median in this year's task (0.600 vs 0.588 B^3+F1).

Although we managed to decrease the gap between our μ-average and B^3+F1 scores on 2012 data, these gaps are again considerable on the 2013 data (eval.: 0.759 vs 0.600, train: 0.794 vs 0.642), indicating the inadequacy of our system to perform (specifically NIL) clustering beyond a crude baseline approach, a task the 2013 data was very much geared toward, and our system is not. Concentrating on our μ-average scores and the NW and WB documents, we do however note that in this regard our system performs on par with obtained results on the 2012 data.

We also still note significant gaps between evaluation and training runs, specifically on the 2012 data, and to a lesser extend on the 2013 data. This is an indication that our system, despite the beneficial effect of our ComboNER system, is still very much dependent on the accuracy of the used NER system(s). To address this issue, we contemplate building a system that would combine NER and NED, with both problems providing feedback to each other, in order to obtain a more robust and accurate system with regard to both tasks.

References

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

L. Ratinov and D. Roth. 2009. *Design Challenges and Misconceptions in Named Entity Recognition*. Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), pp. 147-155

L. Mertens, T. Demeester, J. Deleu, P. Demeester and C. Develder. 2012. *UGent Participation in the TAC 2012 Entity-Linking Task* Proceedings of the Fifth Text Analysis Conference (TAC 2012)