The TALP participation at TAC-KBP 2012

E. Gonzàlez[2], H. Rodríguez[1], J. Turmo[1], P.R. Comas[1], A. Naderi[1], A. Ageno[1], E. Sapena[1], M. Vila[3] and M.A. Martí[3]
[1] TALP Research Center, UPC, Spain.
[2] TALP Research Center, UPC, Spain. Now at Google.
[3] CLiC, Universitat de Barcelona, Spain.
{egonzalez,horacio,turmo}@lsi.upc.edu
{pcomas,anaderi,ageno,esapena}@lsi.upc.edu
{marta.vila,amarti}@ub.edu

Abstract

This document describes the work performed by the Universitat Politècnica de Catalunya (UPC) in its first participation at TAC-KBP 2012 in both the Entity Linking and the Slot Filling tasks.

1 Introduction

Both Entity Linking (EL) and Slot Filling (SF) tasks aim at extracting useful information in order to enrich a knowledge base. This document describes the work carried out by the TALP research group of the Universitat Politècnica de Catalunya in its first participation at TAC-KBP 2012 in both the Entity Linking and the Slot Filling tasks for English. The purpose of this first participation has been mainly exploratory, aiming at performing a preliminary assessment of our approaches (one for EL, two different ones for SF) and drawing conclusions on how to improve them.

EL is the task of referring a Named Entity mention to the unique entry within a reference knowledge base (KB). TAC-KBP track defines the task of EL as follows: having a set of queries, each one consisting of a target name string along with a background document in which the target name string can be found and a source document collection from which systems can learn, the EL system is required to select the appropriate KB entry. Queries generally consist of the same name string from different docids. The system is expected to distinguish the ambiguous names (e.g., Barcelona could refer to the sport team, the university, city, state, or person). In TAC-KBP 2012, we have sent one run and evaluated our EL system just for Mono-lingual Entity Linking. The run did not access the web and also did not use query offsets during the evaluation.

In the SF task, the given set of queries is a set of entity KB nodes that must be augmented by extracting all the new learnable slot values for the entity as found in a large corpus of documents. SF involves mining information from the documents and therefore applies Information Extraction (IE) techniques. We have only participated in the English Mono-lingual Slot Filling task, submitting two runs. Both runs differ in the IE approach employed to detect possible query slot fillers in the candidate documents. The first approach is supervised (based on distant learning), whereas the second one is completely unsupervised (based on minority clustering).

The rest of the document is structured as follows. Section 2 describes the query preprocessing step, shared by all the systems. Section 3 is devoted to our Entity Linking approach. In section 4 we describe our Slot Filling approaches, including the shared document preprocessing step and the two different IE approaches applied. Finally, section 5 presents and analyses the results obtained in KBP 2012 by our approaches in both tasks.

2 Query preprocessing

Query preprocessing consists of the following tasks:

- For both SF and EL, a crucial point is generating the set of alternate names, A, for the query entity. For generating A we have used 4 sources of information: The query name (either a word or a multiword) and its type, the available structured information and textual (non structured) information from documents supporting the query.

- For EL, classifying the query entity into the appropriate query type (PER, ORG or GPE) using the Stanford NERC[1]), over the reference document attached to the query. For SF this process is not needed because the type of the entity is known.
- For SF, obtaining, when existing, the corresponding node in KB. The facts associated to this node are retrieved.
- For both SF and EL, we look at Wikipedia (WP) for the possible existence of the corresponding page. Disambiguation pages are discarded. If some infobox is found their slots and values are retrieved.
- For EL, if the type is GPE we look at geographic gazetteers (GNIS[2] and GEONAMES[3]) and select the corresponding entries.

The documents we use as knowledge sources are:

- The reference document attached to the query.
- For SF, when a KB node is included in the query, the attached description document, if existing, and the facts associated to this node when containing free text.
- For both SF and EL when a WP page exists. the textual content of the page is selected.

Using all these knowledge sources, our way of building set A is the following: The set is initialized with the query name scored with 1. Then a set of enrichment procedures are iteratively applied until no more alternate names are found. The new alternate names are scored decresingly. There are two types of procedures for generating altrenate names: generic and type-specific. Generic procedures are the following:

[1]http://nlp.stanford.edu/software/CRF-NER.shtml

[2]http://geonames.usgs.gov/geonames/stategaz

[3]http://www.geonames.org

Query	Name	Alternate names	#
SF558	Barbara Boxer	1.0 Barbara Levy Boxer 0.8 Barbara L. Boxer 0.64 B. L. Boxer 0.56 B. Boxer 0.49 Boxer ...	37
SF520	Hong Kong Disneyland	1.0 Hong Kong Disneyland 1.0 HKDL 0.8 H. Kong Disneyland 0.8 Hong K. Disneyland 0.7 Hong Disneyland ...	30

Table 1: Examples of alternate names

1. We select a set of pairs (*WP infobox, slot*) where slot refers to an alternate name (e.g. *formal name*, *alias*, *nickname*, *also_known_as*, etc.). If we have a WP page we extract these values and insert them into A also with the maximum score. We proceed in the same way with the KB nodes using in this case available facts.
2. We apply the SF corresponding to the generic slot *alternate_name* existing for both PER and ORG, as described in section 4.

Specific procedures, applied iteratively over all the current members of A:

1. For PER. We use a DCG grammar of English person names for extracting the structure of a complex name. For instance, from *Paul Auster* our aim is to detect that the first name is *Paul* and the family (main) name is *Auster*. We then generate valid variants of the original name always preserving the family name. These variants are scored accordingly with the generalization degree, in our example: (*P. Auster*, 0.8), (*Auster*, 0.6).
2. For ORG. We have developed a set of 12 acronym/expansion mapping functions owning credibility scores which can be applied in the two directions:
    - Starting from an acronym we look up in the textual description for the occurrence of valid expansions applying our mapping

functions. We score the valid variants with the credibility of the applied function.

- Starting from a complete form we perform acronym detection equally scored.
- New forms of ORG names can be found removing common company suffixes (e.g. *Inc*, *Company*, etc.).

3. For GPE we extract all the variants existing in the geographic gazetteers and score them with the edit distance between the original form and the variant.

Some examples of alternate names generated with this procedure are shown in Table 1.

3 Entity Linking

Our approach is inspired by recent works on EL using graph-based methods such as (Guo et al., 2011; Hachey et al., 2011; Han et al., 2011). It consists of three steps for each query. Briefly, given a query, we start by selecting those KB nodes which are candidates to be the correct entity for the query (candidate generation step). Then, we create a graph with the selected candidates and information related to them (graph generation step). Finally, we explore the graph relations for ranking the candidates in order to select the most appropriate one for the query (graph-based ranking step).

The rest of this section describes our methods for candidates generation and graph generation, as well as the graph-based ranking approach.

3.1 Candidate Generation

As KB usually contains a large number of entries, it is desirable to avoid brute force comparisions between a particular query and all KB entries and to reduce the search space of potential candidates. Our priority, however, is to generate a large candidate set instead of a smaller one in order to increase recall (McNamee et al., 2010; Lehmann et al., 2010).

In order to get the set of candidates for a particular query, q, our system performs two steps. First, the query is preprocessed using the procedure described in Section 2. So, q is classified as PER, ORG or GPE, and the set A of alternate names for the query name, m, is obtained. Then, the set of candidates, C, for q is retrieved from the KB being each $c_i \in C$ an entry corresponding to one of the alternate name.

3.2 Graph Generation

From C, we create a graph to represent knowledge related to the candidates, which will be useful for later selecting the most appropriate one for q. We can describe the graph we use as follows:

> *The directed graph* ***G****=(V, E), where the vertices set* ***V*** *contains nodes representing all the candidates in C, the query, and the property values for the candidates and the query; and the directed edges set* ***E*** *consists of all weighted labelled connections between the vertices.*

The graph is initialized to a set of disjoined nodes corresponding to the elements of C. To enrich the graph, we need to retrieve the informative parts of each candidate from the KB entry: the set of facts and the wikitext if it exists. In the case of facts, considering each one as a property with a particular value, the property is represented as the label of a directed edge in the graph, whilst the value is represented as a node connected by the edge from the candidate. In the case of wikitext, we extract all NE mentions of types PER, ORG, LOC and MISC from the first 30 tokens.[4] Here, we consider that the most relevant information related to the candidate in the wikitext is frequently described in the first part of the text. Each extracted NE is represented as a node connected with an unlabeled edge to the candidate.

Moreover, we also represent the query in the graph by including a new node, q. Then, we take all NEs occurring within the context of all sentences of the background document in which the query name occurs. These NEs are represented as new nodes in the graph connected to the query node by an unlabeled edge.

An example of a graph generated for the query related to "*Picasso*" is depicted in Figure 1. Candidates for this query are "*Pablo Picasso*" a Spanish painter, "*Paloma Picasso*" a fashion designer and the youngest daughter of "*Pablo Picasso*," and "*Francisco Picasso*" an Olympic and national-record holding swimmer from Uruguay. Some properties of the first candidate are *Place of Birth = Málaga, Spain* and *Children = Paloma Picasso*.

[4]The same NERC is used (Stanford NERC).

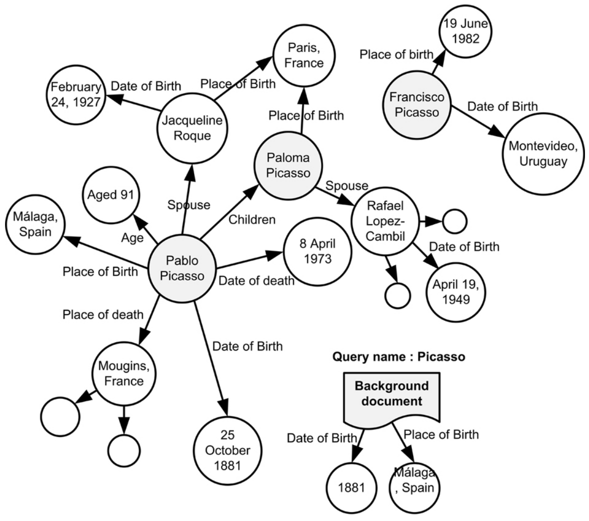


Figure 1: A graph for query name *Picasso*

Note that in the second case the relation is between two candidates.

All edges have a weight which represents the degree of dependency assigned to them. It is used to model and measure the connectivity between nodes. We have manually set these weights as follows. The weights of edges obtained from KB facts are set to 20, which is the highest weight, as we considered them as true information. The weights of those acquired from the candidate wikitext are set to 5. Moreover, the weights of edges related to the query are set to 1.

3.3 Graph-Based Ranking

Given the graph G, the system has to select the correct candidate as the KB reference of q. We score all the candidates by comparing their similarity/relatedness with the query node and select the one having the highest score.

Consider that $C = (c_1, c_2, \ldots, c_n)$ is the set of candidate nodes, q is the query node in the graph G, $P_{c_k} = (P^1_{c_k}, P^2_{c_k}, \ldots, P^m_{c_k})$ is the set of paths between q and c_k without considering direction of edges, where each $P^i_{c_k}$ is represented by the sequence of weights corresponding to the edges in the path, $P^i_{c_k} = \langle w_1, w_2, \ldots, w_r \rangle$, and s_{c_k} is the score of the candidate node c_k, then:

$$s_{c_k} = \begin{cases} \sum_{P^i_{c_k} \in P_{c_k}} \sum_{w_j \in P^i_{c_k}} w_j & \text{if } P_{c_k} \neq \emptyset \\ 0 & \text{if } P_{c_k} = \emptyset \end{cases} \tag{1}$$

Assuming m_q as the query name and $S = \{s_{c_k}\}$, the link between m_q and KB is obtained as follows:

$$link(m_q) = \begin{cases} c & \text{if } \exists c \in C : s_c = \max(S) \geq \beta \\ \text{NIL} & \textit{otherwise} \end{cases} \tag{2}$$

where, β is a threshold, different for each query type estimated using the first 100 queries of KBP 2011 manually tagged.

Figure 2 shows a sample graph structure. As shown in this figure, consider three candidates $C = (c_1, c_2, c_3)$ for a particular query q in the graph. Each candidate is connected to their corresponding properties by the directed edges. Each edge has an assigned weight, w. The initial score for the candidates, s_{c_1}, s_{c_2} and s_{c_3}, is 0 and the initial one for the query, s_q, is 1. Then, each candidate is scored by the products of s_q and the sum of weights for all paths from q to the candidate. In the example:

$$\begin{aligned} s_{c_1} &= s_q \cdot (w^2_q + w^1_{c_1}) \\ s_{c_2} &= s_q \cdot (w^2_q + w^1_{c_2}) + s_q \cdot (w^1_q + w^2_{c_2}) \\ s_{c_3} &= 0, \end{aligned} \tag{3}$$

where w^j_i stands for the weight of the j-th edge from node i.

We select the best scored candidate and return it as our solution if the score is over the threshold β. Otherwise the result is NIL.

In the case of NIL, sometimes, several EL queries refer to the same non-KB (NIL) entity. In these cases, these queries should be collected into one identifiable NIL cluster. For NIL clustering, those queries belonging to the same cluster take the same NIL id in the form of *NILxxxx* being *xxxx* a natural number. The method that we apply for NIL clustering is similar to the approach for ranking candidates.

If query q in the graph G results NIL, then we create a NIL graph (G_{NIL}) that represents several clusters, each one including previous queries related to the same non-KB entity. Each cluster is represented just with its first NIL query (i.e. the medoid) and its

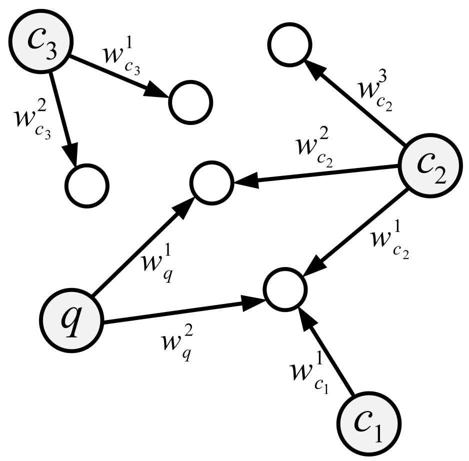


Figure 2: A sample view of our graph structure

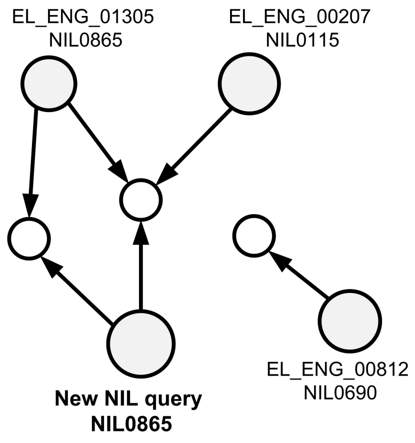


Figure 3: Sample NIL clustering graph

properties. The goal of our NIL clustering method is to select the cluster (i.e., the medoid) to which the query belongs and to assign the corresponding NIL id.

In Figure 3, we show a sample NIL clustering graph, (G_{NIL}). This graph contains three clusters. The medoid of each cluster is labeled by both the query id and the NIL id. As depicted in this figure, these medoids are "EL_ENG_01305," "EL_ENG_00207" and "EL_ENG_00812." These nodes are linked to their corresponding properties. Additionally, a NIL query is temporarily joined to G_{NIL} to infer the cluster to which NIL query belongs. The NIL query has two distinct paths to "EL_ENG_01305" and one path to "EL_ENG_00207."

In order to generate G_{NIL}, we use the same procedure as the one used for generating G when using the textual information. However, in this case, the weights of the resulting edges are set to 1 given that all the properties are extracted from the background documents.

Then, we proceed to find the most appropriate medoid for the NIL query node. This is performed using Equations 1 and 2 with medoids as candidates, C, and $\beta_{\emptyset}$ as threshold. If the NIL query node is linked to a medoid following Equation 2, then the id of the medoid is assigned to the NIL query and the NIL query node is deleted. Otherwise, a new id is assigned the NIL query and the NIL query node is joined to G_{NIL} as the medoid of a new NIL cluster.

In Figure 3, if the score of the node labelled "EL_ENG_01305" is greater than $\beta_{\emptyset}$, its NIL id ("NIL0865") is taken for the query result and NIL query is eliminated from G_{NIL}.

4 Slot Filling task

The UPC system for Slot Filling consists in three steps: 1) preprocessing the document collection in order to collect those documents relevant for each query, 2) applying Information Extraction (IE) patterns to the relevant documents to achieve possible fillers for the slots required for each query, and 3) integrating the resulting slot fillers into the KB knowledge base by normalising extracted fillers (i.e., selecting the most specific fillers under subsumtion for a particular slot, and normalising dates).

We have developed two different IE pattern learning approaches for our exploratory participation in KBP 2012: the first approach based on distant learning and the second one based on unsupervised learning. The rest of this section describes the preprocessing of the document collection as well as both learning approaches.

4.1 Document preprocessing

Prior to evaluation of KBP 2012, the document collection was indexed using Lucene[5] by all the words occurring in the documents.

At evaluation time, a preprocess has been performed in two steps for each query. The first step

[5] `http://lucene.apache.org/`

consists in retrieving the set D of documents containing at least one alternate name of the query expanded as described in Section 2 for the SF task. However, given the ambiguity of proper names, some of the retrieved documents could be related to a real entity different to the required one (e.g., retrieving documents related to Paul Watson -the environmental activist- can result with some documents related to other Paul Watson -the writer, the film maker, and so on-). This is why the second step of the preprocess consists in selecting the set $\hat{D} \subset D$ of documents really relevant for the query.

In order to obtain $\hat{D}$ from D, a particular relevance-feedback approach is performed. This approach is based on the assumption that lemmas frequently found in the close context to an occurrence of a NE can be useful to disambiguate it. The procedure starts preprocessing all the documents in D to get lemmas and POS tags of all words, as well as to detect NE occurrences. The initialization step consists of:

$L = \emptyset$
$\hat{D} = \{d_q\}$, the query reference document

Then, the following steps are iteratively performed:

1. Grow the set L of contextual lemmas[6] for all the alternate names of the query occurring as NEs in documents belonging to $\hat{D}$.

2. Select the subset $K \subseteq L$ of the most relevant contextual lemmas as described below.

3. Grow $\hat{D}$ with those documents from D in which at least one lemma belonging to K occurs within the context of an alternate name.

4. Repeat from step 1 until $\hat{D}$ does not change.

Set K is obtained in two steps. First, L is sorted by score s_i^* as follows:

$$s_i^* = \frac{s_i - \min_j s_j}{\max_j s_j - \min_j s_j}$$
$$s_i = \log \frac{F(l_i)}{f(l_i)} \cdot \frac{f(l_i)}{\sum f(l_j)}$$

[6]We use a centered window of 5 noun, verb or adjective lemmas to the left/right of each alternate name occurrence.

where $1 \leq i, j \leq |L|$, $F(l_i)$ is the frequency of lemma l_i in D and $f(l_i)$ is the frequency of lemma l_i when it occurs as contextual lemma in $\hat{D}$. Then, the minimun set of lemmas $\{l_i\} \subset L$ with greater score is automatically selected as K. Intuitivelly, this can be approached by selecting as threshold l_{th} that l_i supporting the maximum convexity of the curve defined by sorting set L by score s_i^*. This can be computed using the following equation:

$$l_{th} = argmin_i \sqrt{s_i^{*2} - (i/\max i)^2}$$

where $1 \leq i \leq |L|$.

4.2 Distant-Learning Approach

Our first run in the SF task of KBP 2012 follows the distant learning (DL) paradigm for Relation Extraction (RE). DL was initially proposed as a RE approach by (Mintz et al., 2009) and applied to the SF task in preceeding KBP contests by several groups such as (Agirre et al., 2009; Surdeanu et al., 2010; Garrido et al., 2011). DL uses supervised learning but the supervision is not provided by manual annotation but from the occurrence of positive training instances in a KS or reference corpus. In the first proposal, (Mintz et al., 2009) used Freebase, an online database of structured semantic data, as KS. In subsequent applications, Wikipedia (WP) infoboxes have been preferred due to its better precision, at a cost of a drop in recall. In our case we have chosen WP too. Our distant learning approach to the task consisted of the following steps:

1. From a local copy of the English WP,[7] we automatically locate the set of pages corresponding to PER and the corresponding to ORG. For doing so we used the links between WP pages and WP categories as well as the graph structure of WP categories. Let *PagesPER* and *PagesORG* be these sets.

2. We used the mapping between the generic slots and the specific slots occurring in WP infoboxes provided by the organization. Table 2 shows, as an example, the set of specific slots corresponding to the generic slot

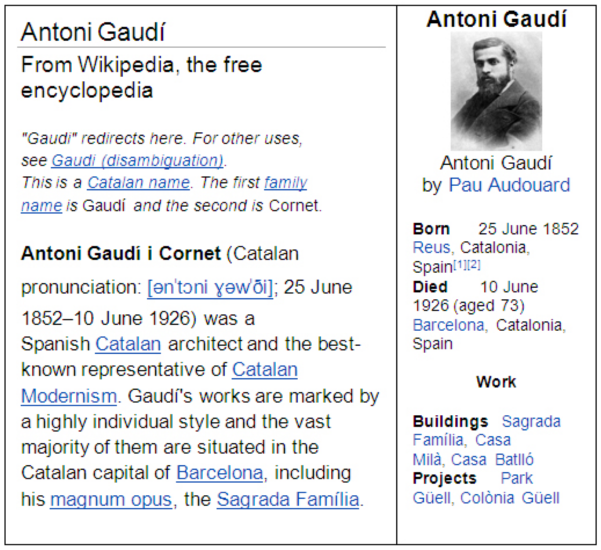
[7]`http://en.wikipedia.org/wiki/English_Wikipedia`. We use for this purpose the JWPK software by Iryna Gurevich: `http://www.ukp.tu-darmstadt.de/software/jwpl`

full_name	nickname	full name
othername	full name	name
burthname	pseudonym	nicknames
othername(s)	name	alias
native_name	playername	fullname
birth_name	birth name	stage/screen
aliases	subject_name	name
other_names	alias	other names
birthname	birth_name	realname
othernames	othername(s)	names
also known as	nickname	

Table 2: Specific slots for the generic slot *per:alternate_names*

per:alternate_names. As shown in Figure 4, WP pages can include both structured (infoboxes, itemized lists...) and unstructured material (text). We took advantage of page infoboxes and page textual content. For all the pages in either *PagesPER* or *PagesORG* we collected all the occurring infoboxes, slots and values resulting in a set of tuples: <*page name, generic slot, infobox name, specific slot, slot value*>. Let *PagesSlotsValuesPER* and *PagesSlotsValuesORG* be these sets. Extracting the values of an specific slot is in some cases easy (e.g. for single-valued slots with a precise type, as *per:date_of_birth*) but in many others it is difficult. In Table 3 some examples of values for the generic slot *per:date_of_death* are shown. Using the Alergia system, (Carrasco and Oncina, 1994), we have learned regular grammars of the slots' values for allowing their extraction. In fact, the number of learned grammars is smaller than the number of slots because some of the values are of the same type, for example the DATE grammar can be used for the slots *date_of_birth* and *date_of_death*.

3. For each of the tuples in *PagesSlotsValuesPER* and *PagesSlotsValuesORG* we extracted the patterns occurring in the text corresponding to the page. For doing so we obtained the possible alternate names of the page name using the same procedure described in section 2. A similar process is carried out for the slot values, for instance for the slot *per:date_of_birth* if the value is *27 April 1945*, also *27-04-1945*, *April 1945*, and *1945* are considered as valid variants (the same grammars used for extraction are used here for generation). As can be seen the process is far to be simple. Two sets of alternate names, *alternateNamesX* and *alternateNamesY* were obtained. We looked on the text for all the occurrences of *alternateNamesX*($X_0, \ldots X_n$) and *alternateNamesY*($Y_0, \ldots Y_m$). For each pair of occurrences (X_i, Y_j) we collected the sequence of words occurring between them and we grouped together all the patterns corresponding to each generic slot. We built in this way the multiset (a set with fre-



Antoni Gaudí

From Wikipedia, the free encyclopedia

"Gaudi" redirects here. For other uses, see Gaudi (disambiguation).
This is a Catalan name. The first family name is Gaudí and the second is Cornet.

Antoni Gaudí i Cornet (Catalan pronunciation: [ənˈtɔni ɣəwˈði]; 25 June 1852–10 June 1926) was a Spanish Catalan architect and the best-known representative of Catalan Modernism. Gaudí's works are marked by a highly individual style and the vast majority of them are situated in the Catalan capital of Barcelona, including his magnum opus, the Sagrada Família.

Antoni Gaudí

Antoni Gaudí by Pau Audouard

Born	25 June 1852 Reus, Catalonia, Spain[1][2]
Died	10 June 1926 (aged 73) Barcelona, Catalonia, Spain

Work

Buildings	Sagrada Família, Casa Milà, Casa Batlló
Projects	Park Güell, Colònia Güell

Figure 4: Example of WP page

Occured Value	Extracted Value
[October 16] , [1952]	October 16, 1952
[March 7] [322 BC]	March 7 322 BC
[748]([Arabian Peninsula])	748
[1368] or [1377]	?
[406 AH] (1015 AD)	1015 AD
25 June , 1274	25 June , 1274
(still alive in 1974)	?
alive	?
'circa' 1126	1126 circa
[1663] (age 23)	1663

Table 3: Examples of values found for the generic slot *per:date_of_death*

quency counts for all the members) *PatternsGenericSlot*. This process resulted in collecting 9,064 patterns for ORG (ranging from 70 for *org:city_of_headquarters*, up to 2,573 for *org:political_religious_affiliation*) and 6,982 patterns for PER (from 23 for *per:cause_of_death* to 588 for *per:title*) with very variable accuracy. In Table 7 some examples of the 57 patterns for the generic slot *per:date_of_birth* are shown.

Once the set of patterns for each generic slot was built (only the most frequent patterns are selected) the process of extraction can be performed as shown in the following steps.

1. For each query we expanded the name onto a set of alternate names containing name variants of the query name (the corresponding *alternateNamesX*).

2. We retrieve from Lucene the documents containing any of the variants in *alternateNamesX*. Some filtering processes were performed in the case of recovering a huge amount of documents (looking only for the more precise variants, e.g., for *John Smith* one of the variants is *Smith* which results on a extremely huge number of mostly irrelevant documents, constraining the search to the whole term *John Smith* could reduce this set to a manageable size, namely, a maximum of 1,000 documents per query.

3. For each query we tried to apply all the patterns corresponding to each generic slot to all the retrieved documents. So if ($X_0, \ldots X_n$) are the variants of the query name and *Patternsgenericslot* contains the patterns of a generic slot we look for the occurrences of an X_i followed by a pattern. The text following this pattern is thus a candidate to be the value of such slot. For locating the right limit of this text we used the same grammars used for extraction in step 2.

4.3 Unsupervised learning approach

Our second approach for learning IE patterns is completely unsupervised from the point of view of using annotated slot-filler examples. Our goal is to explore the approapriateness of using clustering techniques to discover patterns useful to detect relevant relations between pairs of named entity types occurring in text, and then, classifying the relevant relations into the set of possible slots in an unsupervised maner. Following, we describe both the relation detection pattern learning approach and the relation classification approach.

4.3.1 The relation detection approach

For each slot in a template of the KBP scenario of extraction, we can define the pair (t_1, t_2) as the pair of entity types associated to the template itself (t_1 can be ORG or PER) and to the slot (t_2 can be AGE, PER, ORG, CITY, COUNTRY, RELIGION, CHARGE, and so on). For each (t_1, t_2), the procedure starts by gathering the set X of entity pairs, $x_i = (e_1, e_2)$, being t_1 and t_2 the entity types of e_1 and e_2, respectively, and co-occurring in sentences of the document collection. Most of the pairs x_i will not be linked by any particular relation. In fact, a minority of them will be effectively related. In this context, minority clustering can be used to detect groups of related entity pairs as foreground clusters and discard non-related ones as background noise.

Based on these assumptions, our goal in KBP 2012 is to perform initial experiments using the Ensemble Weak minOrity Cluster Scoring (EWOCS) algorithm (Gonzàlez and Turmo, 2012). Concretely, we have used the default configuration to deal with the relation detection task (Gonzàlez and Turmo, 2009; Gonzàlez, 2012), RD-EWOCS up to now, which is briefly described below.

Figure 5 depicts the RD-EWOCS general algorithm. It requires to represent each example as a binary feature vector. The default features used to represent each entity pair $x_i \in X$ are described in Table 4. The algorithm consists in two main steps: the *scoring* of the set of entity pairs related to a particular (t_1, t_2) and the *filtering* of the relevant pairs.

Scoring. Briefly, using an individual weak clustering algorithm f, we randomly produce R clustering models, $\pi = \pi_1, \ldots, \pi_R$ where $R = 100$ by default, from X. The default f for RD-EWOCS is a *Random Bregman Clustering* algorithm, a partition clustering algorithm which consists of the following steps:

- For each clustering model $\pi_c = \{\pi_1^c, \ldots, \pi_k^c\}$ randomly select both the number of clusters

	Feature	**Description**
structural	rightly/lefty	the first NE type t_1 occurs to the right/left of t_2
	dist_X	distance in tokens between the pair is X
	ch_dist_X	distance in chunks between the pair is X
word based	left_X_Y/right_X_Y	token X positions before/after to the left/rightmost NE of the pair has POS Y
	lmid_X_Y/rmid_X_Y	token X positions after/before to the left/rightmost NE of the pair has POS Y
	l_left_X_Y/l_right_X_Y	token X positions before/after to the left/rightmost NE of the pair has lemma Y
	l_lmid_X_Y/l_rmid_X_Y	token X positions after/before to the left/rightmost NE of the pair has lemma Y
	n_left_X/n_right_X	token X positions before/after to the left/rightmost NE is a negative word
	n_lmid_X/n_rmid_X	token X positions after/before to the left/rightmost NE is a negative word
chunk based	ch_left_X_Y/ch_right_X_Y	chunk X positions before/after to that containing the left/rightmost NE of the pair has type Y
	ch_lmid_X_Y/ch_rmid_X_Y	chunk X positions after/before to that containing the left/rightmost NE of the pair has type Y
	chl_left_X_Y/chl_right_X_Y	chunk X positions before/after to that containing the left/rightmost NE of the pair has a head with lemma Y
	chl_lmid_X_Y/chl_rmid_X_Y	chunk X positions after/before to that containing the left/rightmost NE of the pair has a head with lemma Y
	cht_left_X_Y/cht_right_X_Y	chunk X positions before/after to that containing the left/rightmost NE of the pair has a head with POS Y
	cht_lmid_X_Y/cht_rmid_X_Y	chunk X positions after/before to that containing the left/rightmost NE of the pair has a head with POS Y

Table 4: Default feature set for RD-EWOCS

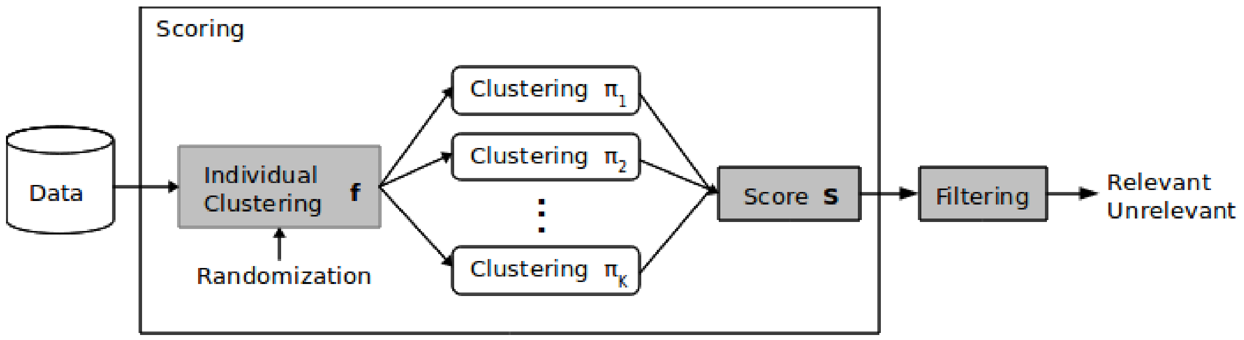


Figure 5: RD-EWOCS general algorithm

$k \in [2, k_{max}]$, where $k_{max} = 50$ by default, and the k seeds, $\{x_1^c, \ldots, x_k^c\}$.

- For each entity pair $x_i \in X$ and cluster $\pi_j^c \in \pi_c$ compute membership grades using a Gaussian-kernel distance as Bregman divergence as follows:

$$grade(x_i, \pi_j^c) = \frac{e^{-D(x_j^c, x_i)}}{\sum_{q=1}^{k} e^{-D(x_q^c, x_i)}}$$

$$D(x, y) = 2\alpha(1 - e^{-\gamma\|x-y\|^2})$$

where, parameters α and γ are automatically tuned in an unsupervised maner with the SOFTBBC-EM algorithm (Gupta and Ghosh, 2006).

- For each cluster $\pi_j^c \in \pi_c$ compute normalized sizes, $size^*$, as the product of the number of non-empty clusters[8], K_c, with the sum of membership grades of all pairs $x_i \in X$:

$$size^*(\pi_j^c) = K_c \cdot size(\pi_j^c)$$
$$size(\pi_j^c) = \sum_{x_i \in X} grade(x_i, \pi_j^c)$$
$$K_c = |\{\pi_j^c | size(\pi_j^c) \geq 1\}|$$

[8]A cluster is non-empty if its size is greater or equal than a threshold. By default, this threshold is 1.

Once π has been computed, each pair x_i is scored as the average of scores s_i^c achieved with each clustering model $\pi_c \in \pi$:

$$s_i^* = \frac{\sum_{\pi_c \in \pi} s_i^c}{R}$$
$$s_i^c = \sum_{\pi_j^c \in \pi_c} grade(x_i, \pi_j^c) \cdot size^*(\pi_j^c)$$

Filtering. Using the same idea as for filtering the most relevant documents in the preprocess (see Section 4.1), the set $\hat{X}$ of those pairs having greater or equal score than the one supporting the maximun convexity of the curve, x_{th} with score s_{th}, is considered as the set of relevant entity pairs:

$$\hat{X} = \{x_i \in X | s_i^* \geq s_{th}\}$$
$$s_{th} = min_i \sqrt{s_i^{*2} - (i/\max i)^2}$$

4.3.2 The relation classification approach

The unsupervised pattern-detection we have described so far, produces a set of entity pairs (e_1, e_2) that are related. But the exact nature and meaning of this relation remains unknown. Thus, we implement an unsupervised classification method that assigns each entity pair to the most likely template slot defined in the KBP evaluation.

Our method comprises two steps: first, the relations are separated according to the entity types (t_1, t_2). For each type pair we do an agglomerative clustering that groups the similar relations into some clusters. Second, we use an unsupervised similarity measure to map each cluster to one of the template slots available for this specific pair of types.

Clustering. To cluster the relation examples, we group them according to the entity types such as (*person*,*date*), (*person*,*location*), (*organization*,*date*), and then perform a clustering of each group. Each example is represented as a binary feature vector, as in the relation detection step. Here we use a subset of the features from Table 4, namely lemmas of tokens and lemmas of chunks in a window of size 5. The clustering algorithm we use is a simple agglomerative clustering with euclidean distance. The hierarchical clustering produces a dendrogram and we use the Calinski criterion (Calinski and Harabasz, 1974) to find an optimum cutting level. A separate clustering is performed for each pair of entity types.

The idea behind this process is to obtain groups of similar relations, ideally, one different relation per cluster.

Mapping. Assuming that each cluster obtained in the previous step corresponds to one different relation, in this step we try to map each cluster to one of the template's suitable slots for that pair (e.g. the pair (*person*,*organization*) can correspond to the slots: *employee_of* and *member_of*). Human experts have selected what t_2 types are the most suitable for each slot.

The mapping is set through an unsupervised process as follows: we take the *description* field (d_s) from the official slots definition document (Ellis, 2012) corresponding to the pair (t_1, t_2). This description is compared to the set of all relation examples in a cluster (each one is a sentence) concatenated in a single text S_s. We compare them using the textual semantic similarity measure of (Corley and Mihalcea, 2005).

This scoring scheme considers the similarity between pairs of words from two text segments, attempting to find for each word the most similar word in the other segment. The similarity between a pair of words is scored using the metric introduced by (Lin, 1998), which takes into account the information content (IC) of each word and their least common subsumer (LCS) in the WordNet taxonomy:

$$sim(v, w) = \frac{2 \cdot IC(LCS(v, w))}{IC(v) + IC(w)}$$

Finally, the word-to-word similarities are combined together into a text-to-text similarity using this function:

$$sim(d_s, S_s) = \frac{\sum_{pos} \sum_{w \in \{d_{s_{pos}}\}} maxSim(w) \cdot idf_w}{\sum_{w \in \{T_{i_{pos}}\}} idf_w}$$

which takes into account part-of-speech tags. This function is directional, we combine both directions by averaging them into a single symmetric similarity measure.

		All	PER	ORG	GPE
All Docs	Overall	0.421	0.599	0.382	0.194
	In-KB	0.311	0.603	0.138	0.192
	NIL	0.545	0.595	0.538	0.203
NW Docs	Overall	0.460	0.620	0.426	0.201
	In-KB	0.344	0.630	0.150	0.197
	NIL	0.582	0.611	0.587	0.232
Web Docs	Overall	0.344	0.533	0.322	0.181
	In-KB	0.253	0.535	0.126	0.183
	NIL	0.461	0.531	0.463	0.169

Table 5: TALP_UPC ML-EL results in TACKBP 2012 (B-cubed+ F-score)

5 Results and Analysis

5.1 Entity Linking

We sent one run for the TACKBP 2012 EL evaluation with following specifications: using wikitext, no access to the Web and without using offset. Table 5 shows our results. It shows B-cubed+ F-scores for both In-KB and NIL queries. Our overall result for all entities and both Newswire (NW) and Web documents is 0.421. We have better score for the PER entity type (0.599) in comparison to ORG (0.382) and GPE (0.194) types. For ORG, one reason is because of difficulty to expand correct forms from acronyms, for instance "ABC" can refer to "American Broadcasting Company" or "Australian Broadcasting Corporation." In the case of GPE, the problem occurs because there are many geopolitical entities with the same name, for instance "Hamilton" may refer to a region in the "New South Wales," "Queensland," "South Australia," "Tasmania" or "Victoria." The results are also better for NW documents in comparison to Web documents. We think that the reason can be the grammar irregularities found in the Web documents.

Analyzing the results shows that we should improve our system in several directions:

- In our run, the pair of offsets for start and end location of the query name in the background document was not used. Then, in the case that for a particular query (e.g., *Hamilton*) two or more different NEs in the background document (e.g., *David Hamilton* and *Daniel Hamilton*) are found, then the offsets are needed to solve the ambiguity.

- When classifying a query, our NERC could not properly identify the query types PER, ORG, or GPE for some queries. This problem caused the generation of irrelevant potential candidates.

- We need to develop an appropriate method to estimate the NIL threshold ($\beta_\emptyset$). In our participation the selection was done *adhoc*.

- We did not take into account the edges labels during the computation of the scores for candidate ranking. For this reason our ranking procedure is not able to discriminate between very similar relations or properties (e.g., *date_of_birth* and *date_of_death*). The lack of this analysis caused a big drop in the EL scores.

- We did not use any external resource such as: 1) The lists of name variation based on hyperlinks and redirects, 2) a particular KB derived from Wikipedia or external corpora to check the correctness of facts or aliases, or 3) a training data set derived from Wikipedia.

From our point of view, the reasons described above do not invalidate the graph-based approach, as most of the recent research devoted to EL explores similar approaches. In this sense, we think that there is room enough to improve our results.

5.2 Slot Filling

Regarding SF, we submited the distant-learning based approach and the unsupervised based one as Run1 and Run2, respectively. Table 6 shows the results achieved.

Run	P	R	F1
Run1	0.224	0.043	0.072
PER	0.241	0.058	0.093
ORG	0.152	0.017	0.031
Run2	0.013	0.005	0.007
PER	0.015	0.002	0.003
ORG	0.012	0.011	0.012

Table 6: TALP_UPC SF results in TAC KBP 2012

For Run1, the statistics of the official results were of 0.04 Recall, 0.22 Precision and 0.072 F1. These results are not bad in terms of Precision (0.11 median) but are very low in terms of Recall (0.08 median). As we do not use any confidence scoring for our answers, NIL is assigned to slots to which no valid assignement has been found. So, for analysing our errors we focus on not NIL answers. For Run2, the results for both types of queries, PER and ORG, are very poor. We achieved 0.005 and 0.016 for Recall and Precision, respectively.

First we present an analysis of the query and document pre-processes, common to both runs.

Regarding both query and document preprocess, a white box evaluation has been carried out taking as a reference V1 of the Assessment Results provided by the organisation. Therefore, we have computed the total recall according to the documents that have been successfully used to extract any correct slot value for any of the slots of the 80 queries (filler judgement column equal to 1).

The recall of the IR phase has been 0.96. A 20% of the documents not found in this step were due to the fact that we failed to include the query names followed by a saxon genitive in the list of alternate names, while 77% of them were due to problems in the generation of the alternate names (missing diminutives, such as "Cathie Black," too general names such as "Arsenal," etc.). Recall for PER queries was 0.98, whilst for ORG queries it was 0.95. Even though this difference is not very significant, we have seen that the generation of alternate names for ORG queries performed worse than for PER queries, due to the less robust methods applied to the task, specially for the case of acronym expansion/compression, as discussed in Section 5.1. The average number of alternate names per query was of 10.5 for PER and 2.9 for ORG.

As to the result of the subsequent process of selection of relevant documents, the recall was 0.83 (0.92 for PER queries, 0.78 for ORG ones). This is partly due to the fact that there were some queries for which, wrongly, just the reference document was found as relevant. The reason for such behaviour was our assumption that alternate names of queries occur as NEs in preprocessed documents. However, this fact strongly depends on the accuracy of the NERC system used. In particular, no alternate name has been recognised as NE for the reference document of some queries with the NERC system we used. As a consequence, the set of keywords useful to retrieve more relevant documents is empty for these queries. This makes our relevant feedback approach stop without providing more documents than the reference ones. Specifically, we discovered that for 13 queries no document other than the reference one was retrieved and for 8 other queries less than 4 document were retrieved. On the other hand, this filtering process turns out to be important as to the reduction in the number of documents: the average number of 1,866 documents found by the IR process is reduced to 611 documents, with an average reduction of 49.94% (56% for PER queries, 43% for ORG ones).

Now, focussing on the analysis of features specific this run, we proceed grouping the results in two axes: queries and slots.

From the queries axe we observe that the distribution of correct answers is extremelly query biased. In fact most of the queries have no answers at all (only 13 from the 40 PER queries and 4 from 40 ORG queries generated some results). This explain our low Recall figures. A second observation is the extremelly unbalanced performance of our system for PER and ORG: 66 correct answers were extracted in top position for PER (0.24 Precision) but only 12 for ORG (0.15 Precision).

Moving to the slot axe we discover that 12 out of the 16 ORG slots produce no results (only 7 for PER). We have manually analyzed a sample of 25 patterns from the pattern sets of all the slots. The results were significant: for PER, all but one (*per:age*) of the slots got an accuracy over 0.9, while for ORG only one slot (*org:alternate_name*) got an accuracy over 0.5.

The reasons why this happens are multiple:

- *PagesORG* are less accurate than *PagesPER* possibly due to the difficulty of obtaining the set of relevant categories for ORG. Getting the relevant WP categories for PER is straigthforward. This is not the case of ORG where categories are spread within the whole set of WP categories.

- Less infoboxes are filled for ORG.

Pattern
was born in
born
on <DATE> in
in
<DATE> in
born in
was born on
was born on <DATE> in
<DATE>
was born
was born and raised in

Table 7: Some of the best scored patterns for the generic slot *per:date_of_birth*

- ORG generic slots mappings are less reliable than PER ones, For many slots the grammars used are really precise (as DATE or PLACE) in the case of PER, but present a great variability in the case of ORG. Locating a PERSON, a DATE or a LOCATION within a value string is easier than locating an ORGANIZATION.

- The patterns extracted for PER are in many cases very short (as shown in Table 7) and frequent. This is not the case for ORG where many patterns are long and occur with very low frequency.

- Most of generic slots for PERSON are single-valued, in the case of ORG the situation is the contrary.

- While persons use to show a similar profile, organizations, present a great variability, for instance a political PARTY or a football TEAM have few points in common.

- Sometimes the mappings between generic and specific slots provided by KBP organizers were not accurate enough. For instance, for *per:age*, the slots contain a large number of varied wordings cointaining the age together with many other useless and noisy information. The grammar learned from this material is obviously extremely unaccurate.

Focusing on Run1, the main reasons why we obtain poor results, besides those presented above for preprocessing steps, are the following:

- According to (Gonzàlez, 2012), EWOCS performance improves if the size of the ensemble of clusterings is selected taking into account the size of the data set, so that large data sets require large ensembles. In this sense, we think that our unsupervised approach requires much more than 100 clusterning models to achieve good results for detecting slot fillers in KBP corpus. This does not unduly penalize the efficiency of the system given that the computation of the clustering models can be paralelized.

- The process of clustering and mapping we have presented in Section 4.3.2 finds the KBP slot that best matches the semantic content of the examples present in each cluster. But due to the high degree of unsupervision, these procedure does neither guarantee that all clusters will be mapped to a different slot, nor that all slots will have an assigned cluster. Additionally, with this method it is not possible to decide that a cluster is not capturing any of the relations expressed by the slots and therefore these examples should be filtered out. This may be a serious drawback in some cases. For example, nothing prevents the system from learning a classifier that splits the relations involving the (*organization,person*) into three clusters and assigns all of them to the *org:shareholders* slot and none to the *org:founded_by* slot.

Taking into account all these points, we think that there is room enough for improvements in both approaches presented to deal with SF task.

Acknowledments

This work has been produced with the support of the project KNOW2 (TIN2009-14715-C04-04).

References

E. Agirre, A. X. Chang, D. S. Jurafsky, C. D. Manning, V. I. Spitkovsky, and E. Yeh. 2009. Stanford-UBC at TAC-KBP. In *Proceedings of the Second Text Analysis Conference (TAC 2009).*

T. Calinski and J. Harabasz. 1974. A dendrite method for cluster analysis. In *Communications in Statistics-theory and Methods*.

R. C. Carrasco and J. Oncina. 1994. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications*. Springer-Verlag.

C. Corley and R. Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, EMSEE '05.

J. Ellis. 2012. *TAC KBP Slots, Version 2.4*. Linguistic Data Consortium.
http://www.nist.gov/tac/2012/KBP/task_guidelines/.

G. Garrido, B. Cabaleiro, A. Pe
nas, A. Rodrigo, and D. Spina. 2011. Distant supervised learning for the TAC-KBP Slot Filling and Temporal Slot Filling Tasks. In *Text Analysis Conference (TAC 2011)*.

E. Gonzàlez and J. Turmo. 2009. Unsupervised relation extraction by massive clustering. In *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM)*.

E. Gonzàlez and J. Turmo. 2012. Unsupervised ensemble minority clustering. In *Research report. Department of Llenguatges i Sistemes Informátics (LSI - UPC)*.

E. Gonzàlez. 2012. *Unsupervised Learning of Relation Detection Patterns*. Ph.D. thesis, UPC Programme in Artificial Intelligence.

Y. Guo, W. Che, T. Liu, and S. Li. 2011. A graph-based method for entity linking. In *5th International Joint Conference on Natural Language Processing*.

G. Gupta and J. Ghosh. 2006. Bregman bubble clustering: A robust , scalable framework for locating multiple, dense regions in data. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*.

B. Hachey, W. Radford, and J. R. Curran. 2011. Graph-based named entity linking with wikipedia. In *Lecture Notes in Computer Science, Volume 6997*.

X. Han, L. Sun, and J. Zhao. 2011. Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th International ACM SIGIRC-conference on Research and Development in Information Retrieval*.

J. Lehmann, S. Monahan, L. Nezda, A. Jung, and Y. Shi. 2010. LCC approaches to knowledge base population at tac 2010. In *Text Analysis Conference*.

D. Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of 15th International Conference on Machine Learning (ICML)*.

P. McNamee, H. T. Dang, H. Simpson, P. Schone, and S. M. Strassel. 2010. An evaluation of technologies for knowledge base population. In *Proceedings of 7th International Conference on Language Resources and Evaluation (LREC)*.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the ACL-AFNLP Joint Conference*. ACL, August.

M. Surdeanu, D. McClosky, J. Tibshirani, J. Bauer, A. X. Chang, V. I. Spitkovsky, and C. D. Manning. 2010. A simple distant supervision approach for the TAC-KBP slot filling task. In *Proceedings of the TAC-KBP 2010 Workshop*.