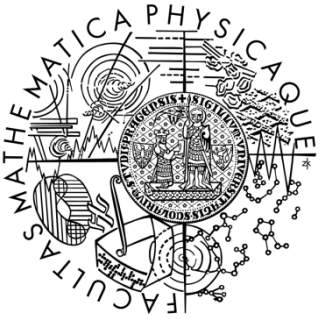
Charles University in Prague

Faculty of Mathematics and Physics

DIPLOMA THESIS



Sergio Raúl Duarte Torres

Entity retrieval on Wikipedia in the scope of gikiCLEF

Institute of Formal and Applied Linguistics

Supervisor: Mgr. Pavel Pecina, Ph.D

Study program: Computer Science

Study specialization: Mathematical Linguistics

European Masters Program in Language and Communication Technologies (LCT)

2009

University of Groningen

Faculty of Arts

MASTER THESIS

Sergio Raúl Duarte Torres

Entity retrieval on Wikipedia in the scope of gikiCLEF

Supervisor: Dr. Gosse Bouma

European Masters Program in Language and Communication Technologies (LCT)

2009

I hereby declare that this diploma thesis is my own work and where it draws on the work of others it is properly cited in the text.

I understand that may thesis may be made available to the public.

Prague. August 7, 2009 Sergio Raúl Duarte Torres

_____

Title: Entity retrieval on Wikipedia in the scope of gikiCLEF

Author: Sergio Raúl Duarte Torres

Department: Institute of Formal and Applied Linguistics

Supervisor: Pavel Pecina, Ph.D, Gosse Bouma, Ph.D

Supervisor's e-mail address: pecina@ufal.mff.cuni.cz, g.bouma@rug.nl

Abstract: This thesis presents a system to retrieve entities specified by a question or description given in natural language, this description indicates the entity type and the properties that the entities need to satisfy. This task is analogous to the one proposed in the GikiCLEF 2009 track. The system is fed with the Spanish Wikipedia Collection of 2008 and every entity is represented by a Wikipage. We propose three novel methods to perform query expansion in the problem of entity retrieval. We also introduce a novel method to employ the English Yago and DBpedia semantic resources to determine the target named entity type; this method is used to improve previous approaches in which the target NE type is based solely on Wikipedia categories. We show that our system obtains promising results when we evaluate its performance in the GikiCLEF 2009 topic list and compare the results with the other participants of the track.

Keywords: entity retrieval, GikiCLEF, semantic resources

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

The increasing amount of digital information available to the public has produced new information needs and changes in the way users search for information. Although search engines have been valuable tools to find relevant documents in large unstructured data sources, more fine grained and detailed information is commonly required by the users. Nowadays, it is common to look for concrete and precise pieces of information to satisfy simple and frequent daily search tasks, as the date of a particular event or the location of a specialized shop. Given the nature of these information requirements, it is highly impractical to search throughout a long list of retrieved documents, as is done in the classic information retrieval scheme. Several approaches have been applied to carry out a more fine-grained search, as question answering and entity retrieval.

1.1 Question Answering

Question answering (QA) systems take as input a question posed in natural language and output a set of potential answers. The research in this area was formalized in the scientific community by the track of Question Answering introduced in the Text Retrieval Conference (TREC) [**1**]. QA systems vary in complexity according to the data resources included in the system, the NLP tools available and the type of questions the system is intended to answer. Nevertheless, a base architecture is found in most of the QA systems. A sketch of this architecture is shown in Figure 1.

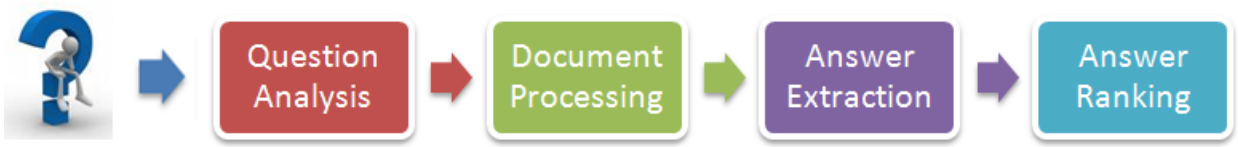


Figure 1. QA basic architecture

First the user's question is analyzed syntactically and semantically using several NLP tools to derive the entity type or information that is required [**2**], with this information a query is constructed to perform an IR search which locates document segments containing a potential answer. In the final phase, the system pinpoints the most likely answer from the set of segments. Generally, the approaches employed to pinpoint the answer are IR or NLP based. In the former the passages are ranked with an IR engine by considering each segment as a document and performing a text search using the query, this approach is used to obtain a small set of candidate answers. In the latter, each segment is parsed using linguistic knowledge to extract semantic interpretations and/or syntactic dependency relations [**3**] which are matched with the relations extracted from the query. This method leads to more accurate answers. Current systems combine both approaches to obtain the fast processing offered by the IR approach with the accurate matching provided by the NLP based methods. [**4**].

1.2 Entity Retrieval

An approach to obtain a more fine-grained search is entity retrieval. The term entity retrieval has been introduced recently in the literature; nonetheless previous research in expert search [**5**] and factoid question answering contemplate tasks directly related to the problem of entity search and ranking. In entity retrieval users are able to obtain a ranked set of named entities as person, locations and dates from any data source. These systems are highly useful to find entities representing concepts related to a given topic or to obtain a set of candidate entities of the same type.

Entity is defined as an object, being or an abstract representation of a concept which has separate existence and can be referenced by a name. Entities can be categorized according their types. Although the number of types is unbounded, in practice a small set of types is defined, for instance in the CoNLL 2002 shared task of language-independent named entity recognition four types are defined: persons, organizations, locations and miscellaneous [**6**]. Ontologies of types expressing type hierarchies and semantic relations between types are also employed, for example the type Scientist can contain the subcategories mathematician, computer scientist, physician and so on.

The tasks that are solved using entity retrieval are mainly of three kinds [**7**]:

(1) Extraction and ranking of types related to a query topic
(2) Search of entities of a defined type
(3) List completion

In the first task, a topic is specified by the user and the system returns a set with the most relevant entity types related with the topic. For example if the topic is *soccer players*, the system should be able to give high scores to entities as *Ronaldo* or *Messi*. Other entities related to soccer players can also be output by the system as *Manchester United* or *Nou Camp*, though the rank of these entities would be lower than the first entities mentioned.

In the second task, the user specifies the topic and the entity type he is looking for. Under this task definition, entities as *Nou Camp* would be considered incorrect for the topic *soccer players*. A version of this task was introduced in the TREC Enterprise track editions 2005 and 2006 [**8**] in which it is required to extract lists of persons who are knowledgeable about a certain topic.

In the third task the user specifies the topic, the entity type required and some entities exemplifying valid answers. In other words given a topic text, a set of entities *S* and a set of example entities *E* in *S*, the system have to return more entities like the ones present in the set *E* and that satisfy the topic text. For instance if the topic is described with the words *cities in the Netherlands* and the user provides in the set of examples the entities *Groningen* and *Rotterdam*, the system should return entities as *Utrecht* or *Den Haag*.

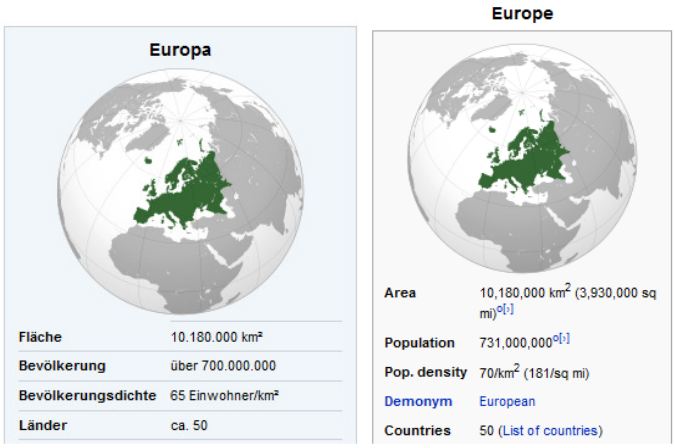
The INEX entity ranking track introduces the task (1) and (3). The track makes use of the Wikipedia data and for the task (3) the Wikipedia categories are used to specify the target entity type. In this track every entity is represented by a Wikipedia article.

1.3 The GikiCLEF task

Recently, a new evaluation task has been introduced under the CLEF multilingual question answering track called GikiCLEF (a previous edition of the same track was named GikiP). The task is to find "Wikipedia articles that answer a particular information need which requires geographical reasoning of some sort" [**9**]. Systems performing well in this task can contribute to exploit the rich information resources offered in Wikipedia by providing a list of articles as answer to complex questions.

This task involves the use of techniques from the fields of document retrieval, question answering and entity retrieval. First, it is necessary to understand the type of information the user is looking for, in other words to obtain the entity type the user wants to retrieve. Furthermore, it is necessary to understand the restrictions that the user imposed on these entities. For example, in the topic "*List Portuguese Pop/Rock groups created in the 90s.*"[1], the entity corresponds to *pop/rock groups* while the restrictions are: *from Portugal* (*or that sing in Portuguese*) and *created in the 90s*. Ideally, systems should perform reasoning on the Wikipedia data and other information domains using the restrictions obtained.

Note that the extraction of the entity types and restrictions from the user question highly resembles the question analysis performed in question answering. Also, this task has a similar definition to the entity retrieval tasks (2) and (3): search of entities of a given type and list completion. In these tasks we have to retrieve a set of entities according to the entity type specified, although in GikiCLEF this information has to be inferred from the text topics. In task (3) a list of example entities is provided along with the topic. These examples can be seen as an abstraction of the restrictions that are expressed in the GikiCLEF topics, since these examples constrain the granularity of the entity type and the characteristics that the entities have to satisfy to be considered valid. There are other interesting features involved in this task such as the presence of polysemy and homonymy phenomena in the Wikipedia collection and the multi-lingual nature of the problem. These features introduce new challenges such as the need to deal with conflicting data of the same entity across languages. This case is exemplified in Figure 2, which shows some discrepancies found in the English and German infoboxes corresponding to the *Europe* entity.



(A) Infobox in German **(B) Infobox in English**

Figure 2: Infoboxes of the entity *Europe* shows discrepancies in the population(bevölkerung) and population density(bevölkerungsdichte) values

[1] Obtained from the GikiCLEF sample topics 2009: http://www.linguateca.pt/GikiCLEF/

Variations in the meaning of entity names also occur across languages. For example, the word *Australia* in English can refer to a country or a continent while in Spanish or Portuguese this word can only refer to the country (the continent is referred as *Oceania*). The multi-lingual aspect of this task also represents an opportunity to retrieve information about highly dependent cultural topics which are only available in specific languages [**9**].

The use of the Wikipedia collection is another motivation to perform research on this task. Wikipedia represents a goldmine of information for researchers given that Wikipedia is currently the largest and most used encyclopedia. The English version of Wikipedia already contains more than 2.9 million articles and contains a large network of links and other semi-structured data as infoboxes and categories, which represents a valuable source to obtain semantic data [**10**]. The quality of the information provided in Wikipedia is also very high given the thousands of contributors writing new documents and judging existing articles. Several research studies make uses of Wikipedia, particularly in the fields of natural language processing, information retrieval and ontology extraction. Medelyan (2008) provides a comprehensive summary of the research studies using Wikipedia.

The evaluation in GikiCLEF is performed using an XML version of a 2008 snapshot of the Wikipedia collection. Fifty topics were released to test the systems. The score function emphasizes the accuracy and rewards multilingual answers. The function is shown in Equation 1.

$$score \quad = c_{en}\frac{c_{en}}{n_{en}} + c_{nl}\frac{c_{nl}}{n_{nl}} + c_{es}\frac{c_{es}}{n_{es}} + \cdots,$$

Equation 1. Score function utilized in GikiCLEF. c_x corresponds to the number of correct answers in language *x*. n_x refers to the total number of answers in language *x*.

1.4 Research objectives

This thesis presents a system to retrieve entities specified by a question or description given in natural language, this description indicates the entity type and the properties that the entities need to satisfy. The system is fed with the Spanish Wikipedia Collection of 2008 and every entity is represented by a Wikipage. This task is analogous to the one proposed in GikiCLEF.

Furthermore, given the multi-lingual nature of the GikiCLEF task, we show methods to exploit semantic resources in English (particularly DBpedia and Yago) to improve the precision and recall of the system. Similarly, we show that exploiting the semi-structured data provided in Wikipedia (such as categories, infoboxes and cross-lingual links) also lead to improvements in the performance of the system.

Our system provides a modular framework to investigate the impact of the methods proposed in the following parts of the system:

- Question analysis: Methods to fetch the entity type and the restrictions specified in the user's question.
- Query expansion: Methods to exploit semantic resources to increase the performance of the system by applying query expansion.

- Generation of candidate entities: Explore mechanisms to generate a set of candidate entities by using an IR engine exploiting the information provided by Wikipedia categories, link structure and other external ontology resources.
- Entity filtering: Methods to filter the entities according to the restrictions imposed by the user.

1.5 Thesis overview

In Chapter 2 we present the most relevant related literature, which correspond to the latest methods developed for the Expert entity extraction and TREC list completion task. Also the approaches explored for GikiP (2008) are considered. Chapter 3 describes the architecture of our system and presents methods to exploit semantic resources in English for systems in foreign languages (Spanish in our case). Chapter 4 shows the results obtained with our baseline systems and the improvements obtained with the use of semantic resources and other methods proposed. Chapter 5 presents the conclusions and future work.

Chapter 2

Related Work

In the following paragraphs we present the most relevant related work for the entity retrieval problem we are considering. We start describing the three systems that participated in the GikiP 2008 edition and then we describe related work for entity ranking approaches developed in the INEX expert finding task.

2.1 Previous work on GikiP

In the GikiP 2008 (the task is analogous to the one defined in GikiCLEF, nonetheless only the Portuguese, English and German Wikipedia collections are considered) [**9**] three participants presented their systems: GIRSA-WP, RENOIR and WikipediaListQA@wlv.

2.1.1 GIRSA-WP

This system was presented by the Intelligent Information and Communication Systems department of the University of Hagen. It merges the results obtained by the GIRSA system [**11**] which performs textual geographical information retrieval (GIR), and the open domain QA system InSicht [**12**].

InSicht system

The InSicht system performs deep syntactic –semantic parsing of the question and the documents to obtain their semantic network representation [**12**]. These semantic representations are also used to construct the answers. This approach differs from other QA systems in which the answers are directly generated from the retrieved documents.

In the document processing phase each document is tokenized and split into sentences. Then, the WOCADI [**13**] parser is used to obtain a semantic network representation of each sentence. WOCADI is a syntactic-semantic parser of German text that utilizes the MultiNet (multilayered extended semantic networks) formalism. This parser is able to obtain full semantic representations of 48.7% (out of 4.884.879) of the sentences in the *Frankfurter Rundschau* corpus [**12**]. WOCADI is also applied in the question processing phase to extract the semantic representation of the user's question. Query expansion is performed when this representation is expanded to equivalent semantic networks using implicational rules. Further equivalent networks are obtained considering the semantic representations of the synonyms and hyponyms of the relevant words in the question. These techniques allow the system to match states with their demonyms (e.g. France -French) and perform logical entailments between related words (e.g. kill-die).

The expanded query network is compared with the semantic representation of the sentences in the collection, several heuristics and network simplifications are performed to speed up this process. Then, the matched semantic networks and the question type are used to generate candidate answers (noun phrases). The set of answer candidates is sorted by giving priority to longer and more frequent answers.

The GIRSA system

The GIRSA (Geographical Information retrieval by semantic annotation) system makes use of location indicators, query decomposition and other functionalities of the InSicht QA system, such as the extraction of the semantic network representations from the question and sentences in the collection.

Location indicators are text fragments that contain geographical information and are processed to obtain normalized location names. Demonyms (e.g. Dutch-The Netherlands), location codes (e.g. postal codes), languages (Portuguese-Portugal), unique entities (e.g. Charles Bridge, Eiffel Tower) and location names (e.g. China) are some examples of location indicators [**11**]. In this technique the tokens in the text are normalized and looked up in a knowledge base that maps location indicators to location names. In GIRSA the knowledge base is created from the Wikipedia official list of state names and other dictionary resources. The location indicators are normalized by removing morphological variations (like inflection), recognizing multi-word names as single terms, extending acronyms and processing prefixes in the tokens to change the scope of the geographical name if it is necessary.

Query decomposition is performed to obtain a subquery including only the geographical information contained in the question. The subquery is submitted to InSicht and the answers returned by the system are included in the original query. This process is done by merging the semantic network representations of the answers with the representation of the original query. For example the query "Whiskey production on the Scottish Islands" leads to the subquery "Name Scottish islands". Answers of this subquery as "Iona" or "Islay" are inserted in the original question leading to the following revised version: "Whiskey production on Iona/Islay". Another decomposition strategy employed in GIRSA consists of exploiting meronymy knowledge of the geographical type of location mentioned in the question. For example the question "Name cities located in the tropics" would lead to the revised query "Name countries/regions located in the tropics".

In GIRSA, documents are preprocessed by getting the base form of the tokens and obtaining normalized location names, which are extracted using the knowledge base. Then, the processed documents are indexed in a database management system that supports the *tf-idf* IR model. Note that an alternative way to process the documents is getting the semantic network representation of their sentences. In this approach documents and queries are processed like in the InSicht system, this method converts the GIR problem to a question answering problem oriented to the geographical domain.

The GIRSA-WP system

In GIRSA-WP a semantic filter is added to ensure the matching of the entity type specified in the topic and the title of the Wikipedia page representing the answer. For this purpose words containing the required entity type are extracted and disambiguated from the topic description. These words and the title of the potential answers are parsed using WOCADI. The parser assigns to these words an ontological sort and a set of semantic features. The ontological sort is a subset of elements of the hierarchy of 45 types defined in WOCADI (e.g. location, object, propriety, etc.). The semantic features define other characteristics from a set of 100 features such as animate, human, information, agent, among others [**14**]. Then the ontological sort and the semantic features belonging to the topic description and the title of the potential answer are matched. If it is not possible to perform the match the Wikipage is discarded.

In GIRSA-WP the documents are indexed per sentences instead of per documents, as is done in the GIRSA system. Additionally, no geographical annotation was performed on the Wikipedia collection, which means that the location indicators method is performed without normalizing and processing the references to geographical names in the wikipages, nonetheless this process is performed on the topic descriptions.

In the GikiP experiments, the topics were processed and sent to InSicht and GIRSA (which are components of GIRSA-WP). The top 1000 results were obtained from the output of GIRSA and their scores were normalized. The maximum score is chosen in case an answer is output by the two systems. The final set of answers is filtered by discarding answers below a threshold and a further filtering is carried out by applying the semantic filter explained before. Multi-lingual answers were obtained by using the cross lingual links available in Wikipedia.

Several runs were submitted varying parameters in the preprocessing phase as the inclusion of stemming, stop-word removal, identification of location names (only for the GikiP topics) and noun compounds. The authors of the system reported that although the sentence indexing approach considered in GIRSA-WP was supposed to increase the precision of the system, it was not effective in this task. They argued that the lack of geographical annotation in the Wikipedia collection reduces the efficiency of the location indicators technique. Similarly they reported that the WOCADI parser ignores important information that is contained in tables and other semi-structured information sources (e.g. infoboxes, categories). Also they stated that the semantic matching approach is still too strict for the IR oriented tasks.

2.1.2 The RENOIR system

RENOIR (Rembrandt's Extended NER On Interactive Retrievals) is an interactive system that performs semantic queries. The system is maintained by Nuno Cardoso (University of Lisbon, Portugal). Semantic queries are intended to capture more precisely the user needs and processed complex queries like "Places where Goethe lived" [**9**].

Semantic queries are very useful to clarify the context of the search and to specify particular search criteria. For example if we are looking for information about the company *PUMA,* the query would be constructed as *ORGANIZATION: PUMA*, which makes clear that we are not looking information about the wild animal. This method also supports the formulation of more sophisticated queries allowing reasoning on the semantic annotations. For example the query *COUNTRY:?LOCAL:Europe* suggest that the user wants to obtain a list of countries located in Europe. These functionalities are possible if the documents in the collection are annotated. The semantic annotation in this case consists of recognizing the named entities in the documents and disambiguating their meaning.

In RENOIR the semantic annotation is performed using the named entity recognizer (NER) REMBRANDT [**15**]. This NER explores the Wikipedia link network and categories to annotate NE in Portuguese and English texts, language-dependent grammatical rules are also applied for the identification of NE. Nine NE categories are supported in REMBRANDT: person, organization, local, timestamp, value, abstraction, thing, masterpiece and event. This system obtained an F-measure value of 0.57 in the full NER task of the second edition of HAREM [**16**], which is a NER evaluation contest of Portuguese language.

In this system, the Wikipedia collection is indexed with the MG4J [**17**] engine. The categories and link information provided in Wikipedia is also stored in MG4J. The MG4J and the REMBRANDT NER are used to offer a set of *query actions*. These query actions are grouped and performed sequentially to construct *query procedures,* which are intended to find the answers to the user questions (or description topics in GikiP).

Query actions

The query actions can refer to retrieval, mapping, annotation or filtering actions. The retrieval actions are used to search Wikipages, the search is performed with a standard term query search (*term search*) or by specifying a Wikipedia category (*category search*). It is also possible to obtain a list of Wikipages that link to a particular Wikipage (*inlinks search*). The three actions are performed automatically.

The mapping actions are used to map a named entity to its corresponding article in the Wikipedia collection. This process is done semi-automatically.

The annotation actions are performed automatically by utilizing the REMBRANDT NER. The system has two types of annotation actions: NE annotation and document to NE mapping. The first action consists in parsing a Wikipage to obtain the named entities contained in the document. The second action is used to map a Wikipage to the most suitable NE type defined in REMBRANDT.

The last type of query actions defined in RENOIR are the filtering actions. These actions are used for: filtering a list of named entities given a named entity type, filter documents by terms checking if the document under evaluation contains the filtering term or not, and filter documents by evaluation which verifies if the NEs of a given type satisfy a set of restrictions. The filtering of document by evaluation is done manually while the filtering of ne by type and the filtering of documents by term are performed automatically. For example, checking if the NEs of location are situated inside a particular region or if the date NEs occur in a given period of time, are typical cases which require the use of filtering actions.

Query procedures

The query procedures are constructed manually according the information asked in each task. The procedures are built in a pipelined fashion. The following list exemplifies a query procedure constructed for the topic "*African capitals with a population of two million inhabitants or more"*:

1. Search category "*African capitals*"→ Output result in $Docs_1$
2. Annotated NE in $Docs_1$ using REMBRANDT → Output result in $Docs'_1$ *(annotated documents)*, NE_1 *(list of ne in each document)*
3. Filter NE_1 by type ∈ *Numerical value* → NE'_1
4. Filter documents by evaluating $NE'_1 > 2{,}000{,}000$ → Output results in $Docs_2$

The first step retrieves a set of wikipages that belong to the Wikipedia category "African capitals", then the REMBRANDT parser is used to identify and annotate the named entities contained in the retrieved documents, the list of named entities of each document is also stored. In step 3 the named entities that don't correspond to numerical types are deleted from the list of entities. In the final step the set of documents is filtered by checking if the value of the NE chosen satisfied the restriction imposed in the original query.

In the construction of procedures several simplifications take place, particularly with the query actions associated to the evaluation of the restrictions that the NEs have to satisfy. This situation arises because this process is frequently performed combining two actions: *filtering of documents by evaluation* which is done manually, *and filtering of documents by NE type*. Although the latter is done automatically, the NE type that the system has to filter is specified in a manual fashion. In the previous example this situation is shown by the selection of the Numerical NE type in step 3. For a person is easier to infer that a numerical type is required to evaluate the restriction about the population of the cities, nonetheless making this decision in an automatic fashion is not trivial.

Similarly in step 4 the evaluation on the numerical value is done by hand. Following this approach it is also assumed that all the numerical NEs are relevant to evaluate the number of inhabitants. Note that it is also possible to obtain scenarios in which the filtering of documents by evaluation requires geographical reasoning which is the case for the query: "Name all wars that occurred on Greek soil".

Also, there are various simplifications in the query actions processing the categories. For instance the Wikipedia category used to obtain the first set of candidate wikipages is selected manually. If the children of the category chosen only refer to subcategory nodes, further processing will take place until Wikipage nodes are reached. The drawback of this method is that the NE type of the subcategories may differ to the NE type of the category. For example the category *William Shakespeare* contains subcategories as *Shakespearean phrases* or *Places associated with William Shakespeare*, which clearly are not associated to the name entity *Person*.

In future versions the designers of RENOIR plan to provide automatic support to all the query actions described above and reduce the number of simplifications by generating the query procedures in an automatic fashion.

2.1.3 The WikipediaListQA@wlv system

This system was presented by Iustin Dornescu who belongs to the Research Group in Computational Linguistics of the University of Wolverhampton (United Kingdom). The system attempts to exploit the information contained in the link structure of Wikipedia and it makes use of the Wikipedia categories. The architecture of the system is based on the following two processes: *domain identification* and *candidate filtering* [**9**].

In the domain identification phase a Wikipedia category containing the potential NE answers is identified. This is done by using the Connexor FDG [**18**] parser to extract noun phrases from the GikiP topics. The first noun phrase extracted is matched with a Wikipedia Category using the Lucene IR engine [**19**]. The query is constructed by applying lexical rules to the noun phrases. For example for the input: "*Spanish mountains*", the lexical rules rewrite the query as: "*mountains in Spain*", "*mountains of Spain*", "*Spain mountains*" and so on. The authors reported that this simple method is effective for most of the GikiP 2008 topics; however some of the categories identified were too broad which was the case of the topic *Name all wars that occurred on Greek soil,* in which the category retrieved was "*Wars*".

In the candidate filtering phase two strategies are applied: entity and factoid filters. The former analyze each Wikipage evaluating if a given entity is contained in the page or it points to this entity with a link. Factoid filters extracts facts from the wikipages and evaluate them according to the criteria specified in

the query. The facts considered in the system were *population*, *nationality*, *height* and *length*. The facts are evaluated by using the relations *greater than*, *in list* and *not in list* on the facts values. The facts are matched in the text by using infobox look-up and regex patterns.

These filters can be used together to capture the restriction indicated in the topic. For example consider the topic "*Portuguese rivers that flow through cities with more than 150,000 inhabitants"*. The first noun phrase corresponds to *Portuguese rivers*. A suitable Wikipedia category to start with is *Rivers of Portugal*. All the wikipages that belong to this category are extracted. Then, a set of entities related to each river is extracted by using the link structure of the articles. This list is filtered to obtain only a list of entities related to locations, since we are interested in evaluating the number of inhabitants of the cities. This filtering is carried out with the entity filter procedure described above. Once the set of filtered entities is obtained, the system tries to verify the restriction on the population using the factoid filters.

Note that the procedure described above involves some of the simplifications mentioned in the RENOIR system. For example in the previous scenario it is assumed that all the location name entities that are extracted with the entity filters are relevant for the evaluation of the restriction, which it is not always true.

Although this system obtained the best accuracy among the three systems described so far, there are some parts of the system that can be further improved. For example in the domain identification phase, it is common to obtain entities of different type when the children of a Wikipedia category are extracted. This situation suggests the need of a procedure to disambiguate NE types to have cleaner candidate entities. Similarly, there are several difficulties in identifying filters in natural language questions.

2.2. Previous work on Entity retrieval (INEX entity ranking track)

Vercoustre, Pehcevski (INRIA institute) and Thom (RMIT University) developed a system to perform the second task of the INEX entity ranking task: list completion [**20**]. Their system exploits the link structure and the categories of Wikipedia. The system attempts to answer the queries by giving greater importance to those wikipages associated to the categories that are somehow close to the categories of the example entities (recall that in this task entity examples are provided along with the description of the topic). Similarly, greater importance is given to wikipages that are pointed to by the examples or located in the same context of the entity examples, where the context refers to the whole Wikipage or parts of it.

The architecture of the system consists of three modules: topic, search and link extraction module. The topic module processes the INEX topic and generates a query for the IR engine. Optionally the example entities are also added to the query. The query constructed is sent to the search module in which the Zettair (2006) IR engine is used to perform a text search on the XML version of the Wikipedia collection. This engine return a set of answers ranked using the Okapi BM25 similarity measure. The link extraction module extracts the link structure of the best *N* rank wikipages.

The wikipages extracted in the previous modules and the link structures are used to compute three similarity measures that are utilized to assign a rank to each page. These similarity measures are merged using a linear combination. The similarity functions are: the LinkRank score, the category similarity function and the full text IR function.

The LinkRank score assigns a weight to a target Wikipage based on the number of links that point to it. This weight is calculated by using the links referring to the target page of the top *N* documents returned by the IR engine. Specifically, the Zettair (2006) score of the referring pages $z(p_r)$, the number of different entity examples in the referring pages $\#ent(p_r)$ and the number of links pointing to the target page $\#links(p_r,t)$ [**20**] are considered to calculate the score as it is shown in Equation 2.

$$S_L(t) = \sum_{r=1}^{N} z(p_r) * g(\#ent(p_r)) * f(\#links(p_r, t))$$

Equation 2: LinkRank score. Where $g(x) = g(x) + 0.5$ and $f(x) = x$.

The functions $g(x)$ is used to allow the cases when there aren't entity examples in the referring pages and $f(x)$ is used to establish that there is always at least one link pointing to the target page *t*.

The category similarity measure establishes a distance between the categories associated with a target page and the categories associated with the entity examples. The similarity measure is defined as the ratio of the intersection of the categories associated with the target Wikipage and the union of the categories associated with the entity examples. The expression of this ratio is shown in equation 3.

$$S_c(t) = \frac{|cat(t) \cap cat(E)|}{|cat(E)|}$$

Equation 3. Category similarity measure. $cat(t)$ refers to the categories associated to the target page *t* and $cat(E)$ the intersection of the categories associated to the entity examples

The global score $S(t)$ assigned to a target Wikipage is calculated by the linear combination shown in Equation 4.

$$S(t) = \propto S_L(t) + \beta S_C(t) + (1 - \alpha - \beta) S_Z(t)$$

Equation 4. Global score function. $S_L(t)$ is LinkRank score, $S_C(t)$ the category similarity measure and $S_Z(t)$ the score returned by Zettair IR engine

The experimentation on the INEX 2007 data and topics showed that the system obtains better performance scores when the three similarity measures are used together. Nonetheless, it was also found that a bigger weight in the category similarity measure lead to the best performance (the best combination found was: $\propto = 0.1, \beta = 0.8$)

Chapter 3

The system

In our system the input topic is parsed to obtain two types of information: The NE type that the user wants to retrieve, and the restrictions or constraints that the NEs need to satisfy. The first type of information is used to obtain a potential set of Wikipedia categories which along with other ontological classifications define the target NE type. The ontological classification is obtained exploiting Yago [**21**], DBpedia [**22**] and WordNet [**23**]. This is done by matching the Yago/DBpedia types to the user's input question and use this information to filter those entities in the potential set that don't correspond to the target type. Yago and DBpedia provide for each Wikipage a set of types which is used in the filtering; however since these information is provided only for the English Wikipedia, we translated the Yago/DBpedia types to Spanish and we made use of the cross-lingual links of Wikipedia to obtain the types of the Spanish wikipages from the English wikipages. More details of this procedure will be given throughout the chapter.

The motivation for using these resources arises from the ambiguity introduced by the Wikipedia categories. Recall that the same Wikipage can be assigned several categories that might refer to different NE types. Similarly, not always the children of a given category are of the same NE type. Thus, we use these ontological resources to disambiguate the type of the wikipages.

A potential set of candidate entities is constructed based on this information (categories and ontological classification) by including the children wikipages of the categories and then filtering the pages that do not match the NE type extracted. In this process the restrictions may also play an important role to refine the granularity of the NE type. Then, the potential set of candidate entities is filtered by matching the restrictions using the Wikipedia infoboxes and the content of the wikipages involved.

This method is effective in topics such as "*Name Romanian writers who were living in USA in 2003*" or "*Which Norwegian musicians were convicted for burning churches?*" In these examples the target NE types correspond to *Romanian writers* and *Norwegian musicians*. The types are directly mapped to the Wikipedia categories of the same name. The restrictions of these examples (*living in USA in 2003* and *convicted for burning churches*) are normalized and evaluated using a text search on the candidate entities obtained from the categories mentioned above.

However, in several topics of the GikiP 2008 and GikiCLEF 2009 track, the Wikipedia categories mapped from the user's query are too broad, which lead to very large candidate entity sets. Furthermore, for some topics it is not even possible to obtain a Wikipedia category. In this situation, it is more convenient to first obtain the wikipages representing the NE associated with the restrictions. For example in the topic "*Name places where Goethe fell in love*", the target NE type is *places* and the restriction on the NEs is *where Goethe fell in love*. Thus, first we fetch the wikipages associated with *Goethe*. Then, we exploit the link structure of Wikipedia to obtain a set of wikipages that point to and are pointed from the *Goethe* Wikipage. This set is filtered to obtain only wikipages that refer to places. Further filtering take place to satisfy the constraint by matching a normalized version of the text "fell in love" in the context

where the links occur and in other semi-structured features of Wikipedia like infoboxes (which summarize the most important factual information of a given entity).
Additionally, we observed that in some topics it is possible to apply the two methods mentioned above: obtaining a candidate set from the Wikipedia categories and the link structure. In this case it is convenient to intersect the sets obtained with the two methods to increase the precision of the results.

The topic "*Name Spanish drivers who have driven in Minardi*" exemplifies this scenario. The category fetched in this example is *Spanish drivers*, which contains about 37 children (using the Spanish data and including one level of subcategories) and the set constructed from the link structure of the Minardi Wikipage contains 60 elements. The intersection of both sets leads to three elements (*Adrián Campos, Fernando Alonso* and *Marc Gené*) which are former Spanish *Minardi* drivers. This method works nicely in the previous example because the restriction is represented semantically by the set intersection. Nonetheless in some cases the topic can involve more complicated restrictions making it necessary to filter the elements after the intersection by matching the other restrictions mentioned in the user's question.

3.1 External libraries and knowledge bases.

In this section we present a brief introduction to the APIS, tools and knowledge bases employed in our system and we describe their interaction with our system.

3.1.1 Apache Lucene

Lucene [**19**] is a high performance text search engine written in JAVA. The library can be used freely under the Apache license. In Lucene the documents are indexed using the Vector-space model and the search can be performed using several query types such as phrase, wildcard, proximity and range matching queries. Lucene also offers fielded searching which is used to search for specific components or parts of the document. For example in the Wikipedia collection we are interested in searching independently on the categories, content and link structure of the wikipages.

Other features include the possibility to sort the documents based on the fields, to search in multiple indexes simultaneously, to apply stemmers in the document processing phase, to create custom rank functions, among others.

We employed Lucene because of this wide variety of features and because it has been widely used in the literature.

3.1.2 Hibernate

Hibernate is an Open source relational-object mapping framework that provides high performance persistence and query services [**24**]. This framework is used to develop persistent classes following the object-oriented paradigm. Thus, objects can be stored along with their relationships (associations, inheritance, polymorphisms, composition and collections). Hibernate also provides an extension to SQL (HQL) which is more convenient to perform queries on an object-oriented design, nonetheless SQL is also supported.

Although the learning curve of Hibernate is not small and it requires some time to become familiar with the framework and exploit its features, we employed it as a persistence model because it will in the long term increase the productivity and maintenance of the persistence layer development. All the implementation details concerning database connections, query optimization and transactions are managed by Hibernate which leads us focus on the business logic of the system, furthermore several caching implementations are provided to increase the performance of the system.

The persistence layer is highly important in our system because it provides an efficient way to integrate several large and heterogeneous knowledge bases (Yago, DBpedia, GeoNames, and Wikipedia) which are utilized to feed the algorithms designed.

3.1.3 OpenNLP

OpenNLP is a community of open source projects related to natural language processing. It hosts several java-based NLP tools to perform sentence detection, tokenization, POS-tagging, chunking, parsing, named entity recognition and co reference resolution. In our system we employed the parser and the POS-tagger for Spanish. Although OpenNLP already provides a Spanish model, we include some modifications in the parsing because we found that the tagging of imperative verbs wasn't accurate and it is commonly found in the GikiCLEF topics. The modification consists of adding a lexical rules.

3.1.4 DBPedia

DBPedia is a community effort to extract information from Wikipedia and interlink this information with other knowledge bases available on the Web [**22**]. The information is made available on the web under the terms of the GNU Free Documentation License. In the 2008 version DBpedia describes more than 2.6 millions things including 213.000 persons, 328.000 places, 36.000 films, 20.000 companies, among others[2].

This project uses the Resource Description Framework as data model. The knowledge base consists of 274.000 RDF triples, 75.000 YAGO categories and several links to images, external web pages and other RDF datasets[1]. This information has been extracted from the English, German, French, Spanish, Italian, Portuguese, Polish, Swedish, Dutch, Japanese, Chinese, Russian, Finnish and Norwegian versions of Wikipedia.

In our system the DBpedia Ontology is used to classify the wikipages according their type. This ontology was hand-generated from the Wikipedia infoboxes and it contains 170 classes and 900 properties. Although the hand-generated mapping does not cover all the possible infoboxes and properties present in the Wikipedia collection, the most frequent infoboxes are included and normalized. The normalization is used to avoid name duplication.

Since the ontology is defined in English we translated the 170 classes to Spanish manually. The entities described in the data correspond to English wikipages classified according their infoboxes. To extract the type annotations provided in DBPedia we make use of the cross-lingual links to obtain the classification of the English wikipages in their equivalent Spanish wikipages.

[2] http://wiki.dbpedia.org/

3.1.5 Yago

Yago is a semantic knowledge base extracted from Wikipedia and WordNet [**21**]. This project started in 2006 and it is maintained by the Database and Information Systems department of the Max Plank Institute for Informatics. Yago contains more than 2 million entities and 20 million facts about these entities represented as RDF triples which are available to the public under the GNU. This information has been manually verified with an accuracy of 95%[3].

Particularly we are utilizing the Yago type's definition which assigns a set of types to each Wikipage. This ontological classification is based on the Wikipedia categories and WordNet, specifically the classes of this ontology correspond to the conceptual categories of Wikipedia and the WordNet synsets. The hierarchy of classes is built using the hyponymy relations of WordNet and the classes extracted from Wikipedia are connected to the higher WordNet classes [**21**].

In the data we found 4930 different types that were translated to Spanish by using the cross-lingual dictionary and then cleaning and completing the results by hand. The Yago and the DBpedia types are used to classify the entities in the Wikipedia collection. We make use of the cross-lingual links to use the Yago classification of the wikipages since these are only available for English.

3.1.6 WordNet

WordNet is a lexical-semantic database for English developed at the Laboratory of Cognitive Science in Princeton [**23**]. Words are grouped into sets of cognitive synonyms called synsets. Each synset represents a concept. Several semantic relations are provided between synsets: holonymy-meronymy, hypernymy-hyyponymy and synonyms-antonyms. In the version 3.0 of WordNet there are 82.215 synsets for 117.798 unique nouns. In our system WordNet is utilized to perform query expansion and to nominalize adjectives and words with other parts of speech present in the user's query.

3.2 Architecture of the system

The system provides a modular architecture that allows the exploration of different solutions to solve the tasks associated with each module. Some of the modules considered in the current system are: question analysis, query processing, document processing and entity filtering. Further modules can be included to provide more sophisticated functionalities, for example entity ranking and semantic visualization.

Figure 3 shows the diagram of the architecture employed in our system. In the following paragraphs we describe in detail each module and the algorithms utilized in the most important processes.

[3] http://www.mpi-inf.mpg.de/

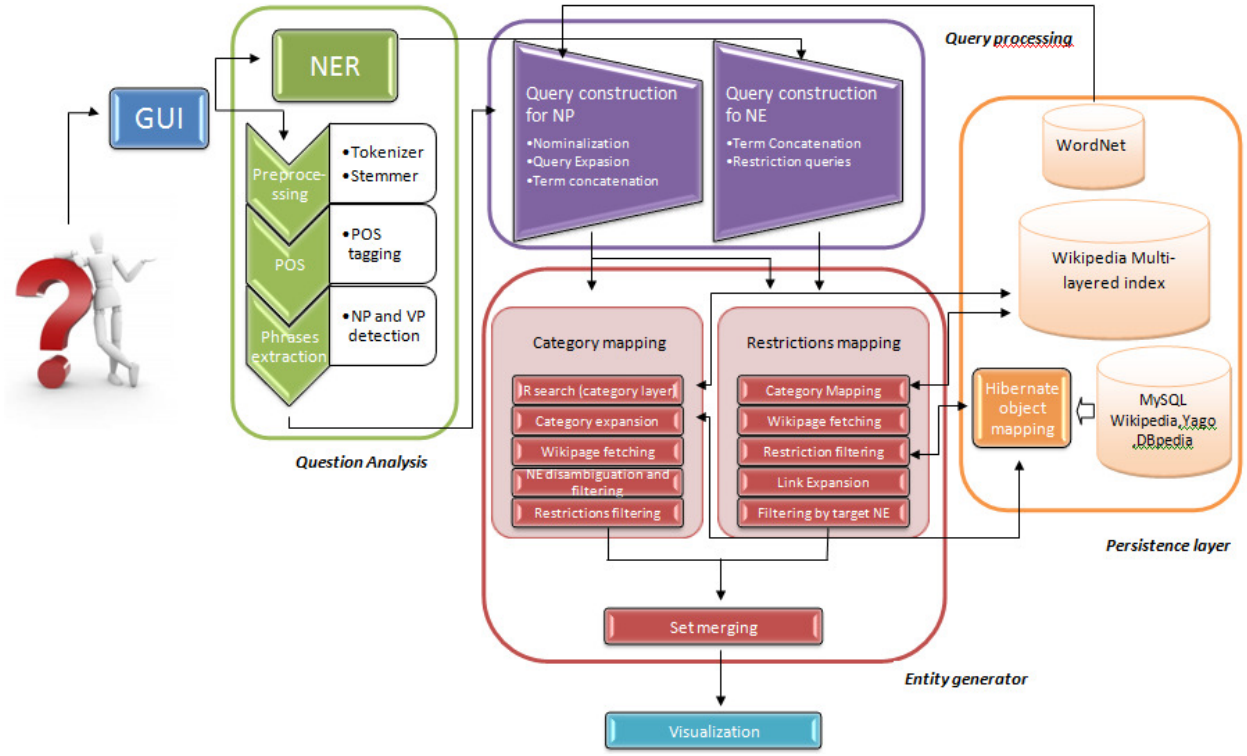


Figure 3: System Architecture

3.2.1 Question Analysis

Tokenization

The tokenization is performed using the OpenNLP Spanish parser. In this process tokens are extracted using regular expressions to recognize whitespaces, punctuation marks, hyphens, numbers, among others. In the Example 1 the input sentence is separated in tokens preserving correctly the format of the numbers.

Input: Poblaciones sobre el círculo polar ártico con más de 100.000 habitantes.
(Places above the Arctic circle with a population larger than 100,000 people)

Output: Poblaciones sobre el círculo polar ártico con más de 100.000 habitantes .

token *1* *2* *3* … *n-2* *n-1* *n*

Example 1. Tokenized output of a sample GikiCLEF topic with *n* tokens

Similarly, Example 2 shows that the question marks are treated as separate tokens and the hyphen is used to split compounded words.

Input: ¿Qué países han ganado un campeonato de Europa de fútbol-sala celebrado en España?
(Which countries have won a futsal European championship played in Spain?)
Output: ¿ Qué países han ganado un campeonato de Europa de fútbol sala celebrado en España ?

token *1* *2* *3* … *n-4* *n-3* ... *n*

Example 2. Tokenized output of a sample sentence with *n* tokens

Word stemming

Stemming is performed to reduce inflected and derived words to their base form. Initially, the Snowball stemmer [**25**] was utilized since it is has been widely employed in the information retrieval literature. Nonetheless, we concluded that the use of this stemmer is inconvenient in our system because the stemmed words obtained often lead to semantic ambiguities. In Example 3 the result of stemming a sample topic with the Snowball stemmer is shown. In this example the word *Novelistas* (Novelists) is stemmed as *Novel*. However if we slightly modify the sample topic to required European novels instead European novelists as is shown in Example 4, we obtain the same output from the stemmer since the word *Novelas* (Novels) is also stemmed as *Novel*. This situation leads to an ambiguity in the recognition of the target NE because it is not possible to distinguish between *Novelistas* (person NE) and *Novelas* (book NE).

Input: Novelistas Europeos románticos y realistas.
(Novelists European romantic and realist)
Output: Novel Europe romantic y realist
Example 3. Stemmed output of the Spanish version of the topic "Romantic and realist European novelists" using Snowball.

Input: Novelas Europeos románticos y realistas.
(Novels European romantic and realist)
Output: Novel Europe romantic y realist
Example 4. Stemmed output of the Spanish version of the topic "Romantic and realist European novels" using Snowball.

For this reason we avoid the use of Snowball and we developed a simple stemmer for Spanish. Two types of morphological variation can be addressed (plurals and gender) given that the information we are extracting is mostly contained in nouns, adjectives and adverbs. Our current stemmer transforms the words in their singular form using a set of rules to perform several suffix replacements. Additionally, accents are deleted after obtaining the singular forms. Although, it would be desirable to normalize the gender of the words in the sentence to increase recall, our stemmer does not include this functionality because changing the gender of a word can lead to a change in its meaning, particularly when this change is performed by applying suffix replacements, for example *mente* (mind) and *menta* (mint).

The set of morphological rules used by the stemmer are summarized in Equation 3. Using these rules on the words *café*s (coffees), *verdades (truths) and pez (fish),* lead to the words *café (coffee), verdad (truth) and peces (fishes).* The notation employed in Equation 3 has the form A/B which means that the suffix A is added to form plural when the word ends in the suffix B. B can also refer to list of suffixes which is represented as $\begin{Bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{Bmatrix}$. In this case a rule of the form A/$\begin{Bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{Bmatrix}$ means that the plural is formed by adding the suffix A when the word ends in the suffixes B_1, B_2.., B_n.

$$\text{P1}\rightarrow \begin{Bmatrix} s/\begin{Bmatrix} \dot{V} \\ \acute{e} \end{Bmatrix} \\ \emptyset/\dot{V}s \\ es/\begin{Bmatrix} C \\ \acute{V} \end{Bmatrix} \\ g/ues \\ z \rightarrow ces \\ c \rightarrow ques \end{Bmatrix}$$

Equation 3. Set of rules to add plural forms. $\dot{V}$ is the set of vowels without accent, $\acute{V}$ is the set of vowels with accent except *é*, and C is the set of consonants except *z*,*g* and *c*. The arrow in last two rules means replacement.

POS Tagging

We employed the OpenNLP tagger for Spanish which uses a maximum entropy model to predict the POS of each word. The model is trained using a collection of news wire articles provided by the Spanish EFE News agency [**26**]. This corpus contains around 270.000 tokens annotated with POS and NE types (four types are distinguished). The data was annotated by the Technical University of Catalonia (UPC) and it was used for the shared task of named entity recognition in the Conference on Computational Natural Language Learning (CoNLL) 2002. The POS tagset is defined by the UPC and it is based on the tagset proposed by the Expert Advisory Group on Language Engineering Standards (EAGLES).

We encountered some inaccuracies tagging the GikiCLEF topics and other simple sentences making it hard to identify phrases using methods based on the POS tags. For this reason a procedure was designed to tune the results and increase the accuracy of the tagger.

First, missing nouns and numbers are detected using the NER parser. Misclassified tags belonging to the closed POS classes such as determiners, conjunctions and prepositions are corrected by looking up a table with a comprehensive list of the words with these POS classes. This list is extracted from the EAGLE's tag definition of the UPC and from Wikipedia. Currently the list contains around 420 items. Further inconsistencies are detected by checking contextual and lexical rules to find sequences of POS that are unfeasible in the Spanish grammar. Although the word order in Spanish is highly free compared to English, there are some combinations of POS that cannot appear together. Some of these rules are summarized in Equation 4. Once an inconsistency is found, a different set of contextual rules is used to

detect the most likely correct tag, these rules are shown in Equation 5 and represent some of the most frequent word order patterns in Spanish simple sentences and questions. Noun is assigned as tag if none of these rules are satisfied. This method is outlined in the Algorithm 1. The notation used in Equations 4 and 5 has the form *A/B*, which means that a match is obtained when the tag or element *B* precedes the tag or element *A*, where *A* and *B* can also represent a set of tags of elements. For example the pair of tagged words *the/A car/B* would be matched with the rule *B/A*.

```
Input: stemmed tokens of the question, NE found in the question
Output: tokens tagged with POS
tagged sentence := tag stemmed tokens using OpenNLP tagger


for each token t in tagged sentence
    if t has an entry in table of Closed POS classes
          Replace POS of t for the one in the table of Close POS
    else if t was tagged as part of a NE
          Replace POS of t for Noun or Number
    else
          if t satisfies any of the rules in the inconsistency rule set
                Find the most suitable POS for t with the fixing rules
                if no rule is satisfied in the fixing rules set,
                      Assign Noun to the t' tag
```

Algorithm 1. POS tagging

$$Inconsistency\ contextual\ rules \rightarrow \left\{ \begin{matrix} Adj/Interrotive\ Pronoun \\ Verb/Det \\ Auxiliary\ Verb/Punctuation \\ A|N|Adj|Adv|Closed\ class\ \ POS\ /Auxiliary\ \ Verb \end{matrix} \right\}$$

Equation 4. Inconsistency rules for POS tagging. Define some of the unfeasible POS order in Spanish

$$Fixing\ contextual\ rules \rightarrow \left\{ \begin{matrix} Noun/Det \\ (Verb/Adj)/Noun \\ (Noun|Det|Prep)/Adverb \\ Verb/Auxiliary\ Verb \end{matrix} \right\}$$

$$Fixing\ lexical\ rules \rightarrow \{Adv\ if\ token\ has\ suffix: \quad 'mente'\}$$

Equation 5. Rules to determine the most likely POS when an inconsistency is found

Though more complex rules can be added in Equation 4 and 5, we found that this method is enough to obtain accurate outputs for simple sentences.

To illustrate how the method works consider Example 5. The output generated by the OpenNLP tagger contains 2 mistakes: *publicado/AQ (published/Adj)* and *Petar/VN (Petar/V).* This first inaccuracy is detected using the contextual rule $A|N|Adj|Adv|Closed\ class\ \ POS\ /Auxiliary\ \ Verb,$ since it is not possible to have adjectives after an auxiliary verb. The correct POS is inferred using the rule $Verb/\ Auxiliary\ \ vebs$, which lead to the replacement of the tag AQ (JJ) to VN (V). This rule says that verbs are commonly found after auxiliary verbs, which is the case of Example 5 because the word located

before the inconsistency found (*published/JJ*) corresponds to an auxiliary verb. The second mistake is corrected using the NER, since *Petar* is part of NE (name of a person in this case),thus we assign a NP tag. Recall that the NE are extracted in the Question Analysis phase and used in Algorith

Input Question:¿En qué países afuera de Bulgaria se han publicado opiniones sobre las ideas de Petar Dunov?
(In which countries outside Bulgaria are there published opinions on Petar Dunov's (Beinsa Duno's) ideas?)

OpenNLP tagging:

¿/FIA En/SPS qué/PT países/NC afuera/RG de/SPS Bulgaria/NP se/PP3 han/VIP
In/Prep which/Dt countries/N outside/Prep from/Prep Bulgaria/N have/Aux been/Aux

publicado/AQ opiniones/NC sobre/SPS las/DA ideas/NC de/SPS Petar/VN Dunov/NP ?/FIT
published/JJ opinions/N about/Prep the/Dt ideas/N of/Prep Petar /VP Dunov /NP

After correction:

¿/FIA En/SPS qué/PT países/NC afuera/RG de/SPS Bulgaria/NP se/PP3 han/VIP
In/Prep which/Det countries/N outside/Prep from/Prep Bulgaria/N have/Aux been/Aux

publicado/VN opiniones/NC sobre/SPS las/DA ideas/NC de/SPS **Petar/NP** Dunov/NP ?/FIT
published/V opinions/N about/Prep the/Det ideas/N of/Prep Petar /NP Dunov /NP

Example 5. Example of output generated by the OpenNLP parser and the tuning method

Phrase Extraction

Commonly the user's question or topic describing the information to be retrieved is expressed in a single sentence. This information can be presented in interrogative or imperative sentences, complex noun phrases are also commonly employed to perform queries in natural language. In all the cases we have to distinguish syntactically the parts of the sentence that refer to the NE type and the parts that specify restrictions or further information about these entities. We found that the NE type is commonly specified in the first noun phrase encountered in the user's question. This is the case for all the GikiCLEF and GikiP topics. Further information concerning these NEs is specified in the noun phrases functioning as object of the sentence or post-modifiers of the main noun phrase. This situation can be seen in Examples 6. In this sentence we have a complex noun phrase with an adjective as pre-modifier and a relative clause as post-modifier. The name entity type is specified in the NP *Escritores alemanes* (German writers) while the noun phrase *ciudados honorarios de Suiza* (honorary citizens in Switzerland) specified the constraint that the target entities have to satisfy.

$[[\text{Escritores}]_{N}$ alemanes $_{Adj}]_{NP}$ $[[[$ que$]_{R_Pro}$ $[\text{sean}_{V}$ $[[[\text{ciudadanos}]_{N}$ $\text{honorarios}_{Adj}]_{NP}]$ $[\text{de}_{Prep}$ $\text{Suiza}_{N}]_{PF}]_{NP}]_{VP}]$ $_{WHNP}]_{NP}$.

$[[\text{German}_{Adj}$ $[\text{writers}]_{N}]_{NP}$ $[[[\text{who}]_{R\_Pro}$ $[\text{are}_{V}$ $[[\text{Honorary}_{Adj}$ $[\text{Citizens}]_{N}]_{NP}$ $[\text{in}_{Prep}$ $\text{Switzerland}_{N}]_{PF}]_{NP}]_{VP}]_{WHNP}]_{NP}$

Example 6. Syntactic structure of the sample topic *Escritores alemanes que sean ciudadanos honorarios de Suiza* (German writers who are Honorary Citizens in Switzerland)

These two noun phrases are frequently separated by relative pronouns such as *that* or *in which.* In more complex sentences the phrases are separated by verb elements.

Thus, first the verb phrases or relative pronouns are detected to split the noun phrases that refer to the NE type and the ones that refer to the restrictions. This is done by matching the tokens with the *PR0* POS (relative pronouns)tag and with a list of common particles used to introduce relative clauses such as *en los cuales* (in which), *en la que* (inside which), *por el que* (along which), *a la que* (whom) and so on. Similarly Verb phrases are extracted by matching the rule *(Aux Verb)* (Verb) (Adv)**.

This process leads to a list of NPs in which the first element defines the NE type. Each one of these NPs is further subdivided to allowing only prepositional phrases as post-modifiers. This further splitting is carried out because the motivation is to match Wikipedia Categories to these NPs and the categories are commonly described by short NPs. The procedure to extract phrases is summarized in the Algorithm 2 and 3.

```
Function splitComplexNP
Input: tokens annotated with POS, tokens[1..n] and pos[1..n]
       Look-up table of dividing phrases: table (in which, along which, etc)
       List of current NP (first NP is mapped as the target NE type)
       List of current VP (VP)

for i:=0 to length(tokens)
    if pos[i] = PR0 or tokens[i..m] ∈ table, for i<m<n
           Insert tokens[0..i] and tokens[j..n] in NP, where j=max(i+1,m)
           return;

    else if pos [i..m] matches (Aux Verb)*(Verb)(Adv)*  for i<m<n
           Removed matched expression from tokens
           Insert matched expression in VP
           Insert tokens[0..i] and tokens[i+1..n] in NP,
           Return;
```

Algorithm 2. Split Complex phrases

```
Procedure PhraseChunker
Input: tokens annotated with POS, tokens[1..n] and pos[1..n]
       Look-up table of dividing phrases: table (in which, along which, etc)
Output: List of NP(first NP is mapped as the target NE type)
        List of VP

#Split sentence in the two main phrases
splitComplexNP(tokens, NP,VP, table)

#NPs are further splitted
for i:=0 to length(NP)
    tokens:= NP[i]
    phraseChunker(tokens, NP,VP,table)
```

Algorithm 3. Phrase Extraction

Example 6 shows the output obtained for the example *Nombre los lugares de Italia que haya visitado Ernest Hemingway a lo largo de su vida (*List the Italian places where Ernest Hemingway visited during his life). First the sentence is split in two noun phrases and one verb phrase. Then each noun phrase is

split further which in this example is only possible for the noun phrase *Ernest Hemingway a lo largo de su vida (Ernest Hemingway during his life)*

First Split:
NP={ {lugares Italia}, {Ernest Hemingway a lo largo de su vida}}
VP={que haya visito}
Further spliting:
NP={ {lugares Italia}, {Ernest Hemingway}, {a lo largo de su vida}}
VP={que haya visito}

Example 6. Phrase extraction for the topic: List the Italian places where Ernest Hemingway visited during his life.

Name entity recognition

We employed the Stanford NER [**27**] and a look up table to detect the NE in each noun phrase extracted. The NER was trained using the Spanish data provided in the Conference on Computational Natural Language Learning (CoNLL-2002). The look up table consists in a list of all the entities (titles of wikipages) available in the collection. Note that we aren't interested in the NE type found with this NER since the types in the system are defined with the Wikipedia categories and the YAGO/DBPedia types. However, some of the algorithms designed to construct the query and generate the candidate set utilize this information as will be shown in the next sections.

3.2.2 Query Processing

This module constructs the queries for the IR Engine based on the noun phrases and named entities extracted in the Question analysis module. Two procedures are considered to construct the queries. In the first method the query is created based on a noun phrase and in the second method the query is created based on a set of NEs. In our system the queries are used to perform three search tasks: mapping of the noun phrases to Wikipedia categories; finding Wikipages associated with these categories and the NEs extracted; and mapping the NP containing the target NE type to the corresponding YAGO and DBPedia ontological sort.

The construction of the queries based on noun phrases follows the procedure described in the Algorithm 4. First, nouns, adjectives and adverbs are included in the query and expanded using the procedure described in Algorithm 5. Then, the tokens that correspond to NE are boosted with a parameter weight. The resulting tokens are concatenated using the syntax specified by the IR engine.

```
Input: Noun phrase np, List of NE in np ne, boosting weight w
Output: String query for IR Engine

for i:=0 to length(np)
       if pos(np[i]) =  N, Adj or Adv
              query[j] := np[i]
              QueryExpansion(query)
              if np[i] ∈ ne
                     Boosts query[j] with w
              j:=j+1
return concatenateTerm(query)
```

Algorithm 4. Query Construction from noun phrases

The query expansion procedure starts with the nominalization of the tokens that are not nouns. This is done by using the redirect links provided in the Wikipedia collection. We found that the redirect links can be exploited to normalize and nominalize entity names. These links are used in Wikipedia to let the user refer to an entity using alternative names and word forms such as: pseudonyms (e.g. Samuel Langhorne Clemens redirects to Marc Twain), abbreviations (e.g. CONMEBOL redirects to the South American Football Confederation), common misspelling forms (e.g. Condoleeza Rice redirects to Condoleezza Rice), alternative spellings (e.g. colour redirects to color) and other adjectival forms (e.g. Peruvian redirects to Peru).

The dictionary of redirect links extracted from the GikiCLEF Wikipedia collection contains 113.790 Spanish entries. Nonetheless missing pairs of country-demonyms were added to the dictionary since this type of information is frequently used in the GikiCLEF topics.

Further expansion takes place using the WordNet lexical database to find synonyms of the terms contained in the input sentence. Given that WordNet contains data solely in English, we make use of the cross-lingual links of Wikipedia to translate the target Spanish word and the results obtained from WordNet. Cross-lingual links are used to direct a Wikipage to its version in a different language, thus the anchor of the links can be used as a very accurate method of translation of NE names. The dictionary (English-Spanish) created from the GikiCLEF Wikipedia contains 112.099 entries. No additional dictionaries were employed. Note that the advantage of using this dictionary instead of other translation resources is that the obtained are associated with a Wikipage while the results obtained by other methods might not be relevant for the search.

```
Input: Query terms q, Redirect Links table redirect, DBPedia Ontology
Output: Extended query terms

for i:=0 to length(q)
       if pos(q[i])<> N
              en := translate_to_English(q[i])

              if q[i] ∈ redirect
                     expand(q[i],redirect(q[i])

              enSYNSETS:= WordnetSynonyms(en);
              spSYNSETS:= translate_to_Spanish(ebSYNSETS)
              expand(q[i], spSYNSETS);

              if q[i] ∈ DBPedia Ontology
              subtypes := subclasses(DBPedia Ontology, q[i])
              expand(q[i], subtypes)
return q
```

Algorithm 5. Query Expansion

Additionally, semantic expansion is carried out when the token analyzed corresponds to an ontological type. This is done by using the DBPedia ontology to include the subclasses of the class mapped to the token. In this ontology the classes and subclasses are related with the hypernymy semantic relation.

Figure 4 shows a part of the taxonomy showing the subclasses of *Actor*. For instance if the user's requires information about german artists, the algorithm expand the terms to include german writers, comedians, actors, etc, since these NE types are also artists.

After the terms are expanded by this procedure, the query is constructed concatenated the terms and specifying the boosting weights, naturally the syntax of this query varies according to the IR Engine used.

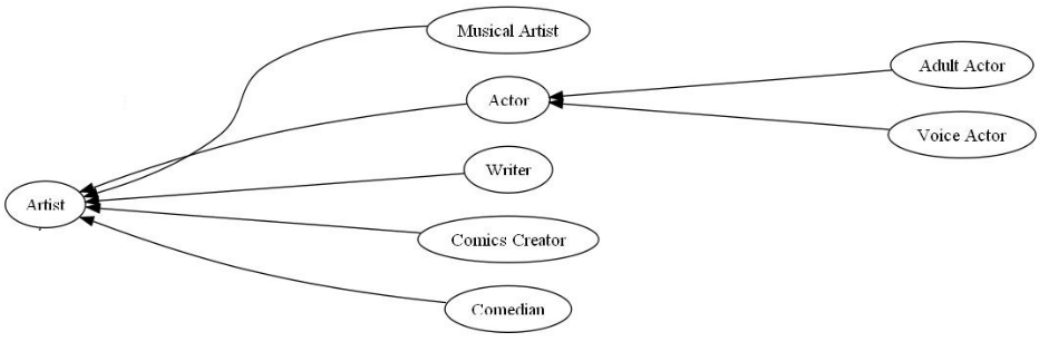


Figure 4. Fragment of DBPedia Ontology [**22**] showing the subclasses of the type Artist.

Example 7 shows the result of applying the query construction mechanisms for the NP *Liste los lugares italianos* (list italian places). First the tokens *Liste* and *de* are ignored because they are not nouns,adjectives or adverbs. Then the token lugares (places) is fetched, since this token is a noun we only apply the synonym and semantic expansion. We found several synonyms for this word such as *sitio* (site) and *posición* (position), however only *posición* (position) is found in the cross-lingual dictionary, leading to the extension (lugar (place) posición (position)). Then, we try to map this token (including the extension) to the ontology, in this case we map the token to the class *place*. Thus all the subclasses of the type *place* are included to the query.

Then, the token *Italiano* (italian) is nominalized using the redirect lookup table, leading to the expansion (Italia italiano). Finally the terms are concatenated following the syntax of Lucene queries. Note that we required the first token to appear in the results besides boosting the group of words containing NEs. We found that this simple modification improves the accuracy of the results.

Liste los lugares italianos
Noun Phrase: Lista/V de/SPS lugares/NC italiano/Adj
Synonyms: lugar spots position (Italia italiano) → (lugar, posición) (Italia, italiano)
Semantic: (lugar, isla, ciudad, localidad, provincia, parque)
Redirect: (lugar, isla, ciudad, localidad, provincia, parque…) (Italia italiano)
Concatenation: +(lugar isla ciudad localidad provincia parque …) (Italia italiano)^{W}

Example 7. Query construction of the NP: *liste los lugares italianos* (List the Italian places)

The queries based on NEs are simply constructed by concatenating the tokens without applying the expansion procedure.

3.2.3 Data Persistence Layer

In this layer the knowledge base and the semantic resources employed in the system is stored. The Lucene IR system is used to index the Wikipages of the GikiCLEF collection and the DBpedia and Yago classes. The IR engine is a highly efficient way to perform text search on a large collection by representing the documents with the vector space model. Additionally, we employed a database to have a scalable repository of ontological resources and to provide an efficient way to access tabular information such as the Wikipedia Infoboxes. Note that the IR engine is not suitable to store and search this type of information because the word order is ignored. The documents in the GikiCLEF collection are an XML version of the original Wikipedia raw data. Several routines were developed on a DOM (Document Object Model) parser to extract the information from the Wikipedia articles and to feed the information to the IR engine and the database. In the following paragraphs we describe the indexes created in Lucene and the structure of the database.

Lucene Indexes

Wikipages index

This index is used to perform text search on wikipages, the articles matched are referred to their database id or title. The text search can be performed on the text content, wikipage title, categories or ontological types (either Yago or DBPedia classes). The categories and ontological types are indexed by concatenating the items associated with each page. Table 1 summarizes the layers and the methods used to process the information. The Stored column refers to the layers in which the original information can be retrieved (e.g. ID, Page Name). The indexed column specifies that the information is stored in an inverted index and can be searched using the vector space model. Note that the original information cannot be restored once is indexed. The Analyzed property indicates that the text information is preprocessed before is stored (tokenization, stop word removal, stemming, etc.). In our system all the indexed documents are analyzed by tokenizing the data, using the stemmer described in section 3.2.1, setting the tokens to lowercase and specifying a stop word list for Spanish (which is provided with the Snowball stemmer).

Field Name (Layer)	Description	Stored	Analyzed	Indexed
Page Name	Name of the wikipage	✓		✓
ID	Database Identifier	✓		
Content	Text content		✓	✓
First Section	Introductory paragraphs of the article		✓	✓
Categories	Wikipedia categories associated to each page	✓	✓	✓
Yago Class EN	Yago classes associated with the article (english)	✓	✓	✓
Yago Class SP	Yago classes associated with the article (spanish)	✓	✓	✓
DBPedia Class EN	DBPedia classes associated with the article (english)	✓	✓	✓
DBPedia Class SP		✓	✓	✓

Links	DBPedia classes associated with the article (spanish)			

Table 1. Layer description of the Wikipage Index.

Category List and Yago/DBPedia class indexes

These two indexes are used to search Wikipedia categories and Yago/DBPedia classes. Each document simply contains the name of a category or a Yago/DBpedia class. These indexes are used as a lookup table to match categories and ontological types with text using the cosine similarity measure. The indexes are queried to map the NE types to the user's query. Although this task can be performed by simpler methods such as hashtable or database lookup, for our needs it is more convenient to employ an IR engine because it is straightforward to use the query expansion techniques and perform queries using natural language phrases.

Name	Description	Stored	Analyzed	Indexed
Yago Class EN	Yago class name English	✓		
Yago Class SP	Yago class name Spanish	✓	✓	✓

Table 2. Yago Classes index

Name	Description	Stored	Analyzed	Indexed
DBPedia Class EN	DBPedia class name English	✓		✓
DBPedia Class SP	DBPedia class name Spanish	✓	✓	✓

Table 3. DBPedia Classes index

Name	Description	Stored	Analyzed	Indexed
Wiki Category name	Category name	✓		✓
Wiki Category ID	Database ID of the category	✓	✓	✓
Wiki Category Stemmed	Category name processed with the Spanish stemmer	✓	✓	✓

Table 4. Wiki Categories index

Database structure

Several tables were designed to store the link structure, categories hierarchy and Wikipedia templates (including infoboxes). The most relevant tables are sketched in Figure 5. In the table *wikipage* the title, name space, name of the file containing the article in the hard disk, internal identifier and Wikipedia identifier are stored. Currently, there are 294.285 (wiki category pages are excluded in this table) wikipages stored in this table. Wikipages can contain several templates, links and categories. The table

wiki_ template stores for each template the name (e.g. Template:Infobox Company), class (e.g: Infobox vcard), identifier and reference to its Wikipage. Templates have a list of <attribute,value> pairs which are stored in *wiki_tablecell*. This table specifies the attribute name, the position of the attribute in the template and the value is a reference to the table *wiki_list*, which is used to store values (strings) associated to the same attribute. Our database contains 379.099 templates and 1.636.137 <attribute.value> pairs.

The table *wiki_link* stores the hyperlink of the item pointed to, the type of link (internal, external, cross-lingual, and redirect link), title (if the link points to a wikipage), text anchor and an internal link identifier. From this table two dictionaries are created: Cross-lingual and Redirect link dictionaries. These dictionaries are loaded in hash tables when the system is initialized.

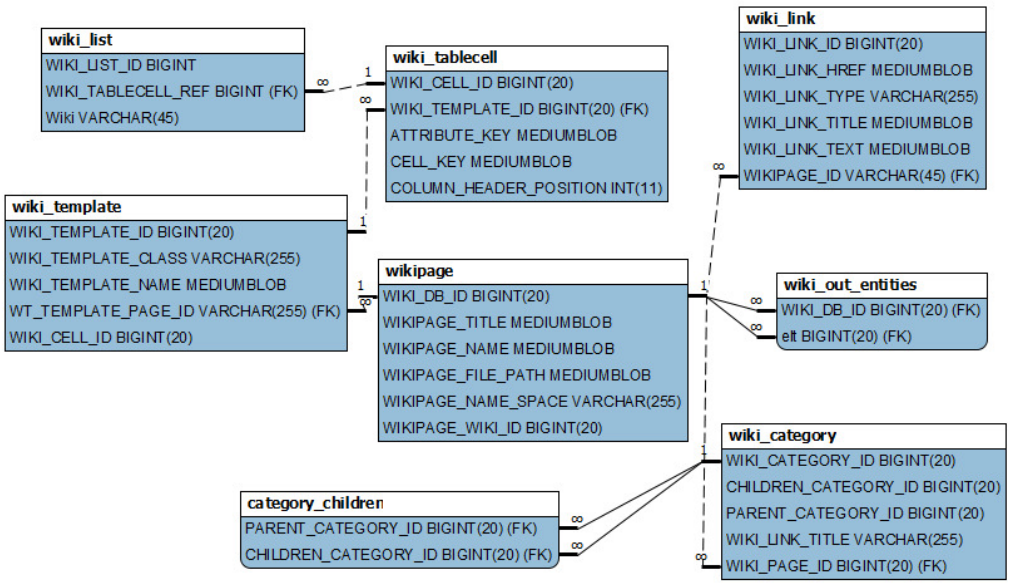


Figure 5. Simplified Scheme of the database

The table *wiki_out_entities* is used to obtain hash tables with the outgoing links and incoming links of each wikipage. The table is built by preprocessing the wikipages to obtain a list of pairs <from WikiId, to WikiId> that specifies the wikipages in the collection that are pointed to from a given Wikipage. The wikipages that point to a target Wikipage (incoming links) correspond to the pairs < *, target ID>. This table contains 1.593.895 pairs.

Finally the wiki_category table contains the list of categories found in the collection. Name, identifier, link used to refer to the category and the wikipage ID that describes the category. These wikipages contain the list of parent categories. This information is used to process a list of pairs <Id category father, Id category child> which allow us to navigate through the Wikipedia category system. However it is important to mention that the GikiCLEF collection does not include description page for all the categories, thus the category structure extracted is not complete. The *wiki_category* table contains 54.487 categories and the *category_children* table contains 45.345. The category structure table is also loaded in a hash table when the system is initialized.

3.2.4 Entity Generator

In this module the entities according the type and restrictions given in the input question are generated. First the NE type is mapped by querying the first noun phrase found in the question to the Category list and Yago/DBPedia classes indexes, thus the target type obtained consists of three sets: Set of Wikipedia categories, Set of Yago classes and set of DBPedia classes. The query is constructed using the procedure described in the Algorithm 4. The results obtained from the IR engine are filtered by choosing only the items scored higher than a threshold, which is usually set between 0.8 and 1 to obtain high accurate results. Additionally, the Wikipedia categories obtained in this method are expanded by including their subcategories. This process is done recursively by setting a parameter to specify the number of levels that are traversed. The subcategories of a target category are extracted from the *category_children* table of the database.

The wikipages that belong to the categories extracted represent the set of candidate entities. Given that wikipages belonging to the same category can belong to different NEs types, we clean the set by ignoring the entities that don't correspond to the target YAGO/DBPedia classes. This is done by adding a filter in the IR engine index of wikipages to ignore the documents retrieved that don't contain any of the target classes in the Yago and DBPedia classes' layers.

To evaluate the restrictions on the named entities, we first map the noun phrases to Wikipedia categories as it was done with the mapping of the NE type. Additionally, we obtain the wikipages associated with the name entities mentioned in these noun phrases. We construct a set of wikipages for each noun phrase adding the members of the categories found and the wikipages of the NE. The phrases that cannot be mapped to categories or don't mention any NE are matched in the text content or infobox of the set of wikipages constructed, this matching is done using the IR engine, for this purpose temporal index is created with the pages of the sets. Pages scored below a threshold are deleted.

In the final step we create a second set of candidate entities by including the entities that point to or are pointed from the entities of the restriction sets. In this set we only include those entities that correspond to the target NE type.

We return as result the intersection of the two candidate entity sets (the one created with the first Noun phrase and the one created with the restriction noun phrases). However, we return one set in case the other one is empty.

The procedure to obtain entities is summarized in the Algorithm 6.

```
Input: List of noun phrases np, Name entities ne
Output:  Set of entities required in input question

queryTargetCat:= constructNPQuery(np[0])
setOfCategories :=
       queryIREngine(queryTargetCat, Categories index,  score_threshold)
setOfCategories := extendCategories(setOfCategories,level)
yagoTargetClasses :=
       queryIREngine(queryTargetCat, Yago index,  score_threshold)
```

```
DBPediaTargetClasses :=
       queryIREngine(queryTargetCat, DBPedia index,  score_threshold)
target_NE_Type :=
       setOfCategories ∪ yagoTargetClasses ∪ DBPediaTargetClasses
candidate_set:=  categoryMembers(setOfCategories)
setA := filterByTargetNE_type(candidate_set)

for i:=1 to length(np)
       queryCat:= constructNPQuery(np[i])
       setOfCategories :=
       queryIREngine(queryCat, Categories index,  score_threshold)
       setOfCategories := extendCategories(setOfCategories,level)

       neQuery:= constructNEQuery(ne[i])
       setOfNE :=
              queryIREngine(neQuery, WikiPages index,  score_threshold)
       restrictionEntities :=  category_members(setOfCategories) ∪
              setOfNe

       if(restricionEntities = ∅)
              add(unSastifiedConstraints,np)

for i:=1 to length(unSastifiedConstraints)
  restrictionEntities :=
       filterEntitiesByConstraint(restrictionEntities
       ,unSastifiedConstraints[i])
expandedEntities := expandByLinkStructure(restrictionEntities, level)
setB := filterByTargetNE_Type(expandedEntities)

if(setA ≠ ∅)
       if(setB ≠ ∅))
              return setA ∩ setB
       return setA
return setB
```

Algorithm 6. Entity Generation

For example in the topic *¿En qué países europeos se suele usar el bidet (*In which European countries is the bidet commonly used?*)* the noun phrase *países europeos* (European countries) is used to obtain the NE type and the noun phrase *el bidet* (the bidet) is used to map the restrictions entities. The query constructed for the first noun phrase is: *países país europeos europeo Europa (countries country europeans Europe). Europa* is obtained by using the redirect links and *país*, *europeo* (country,european) is obtained by using stemming,

This query is used to search on the Category List index. The category obtained is: *Paises de Europa (*European Countries). Then this category is extended to include its subcategories. In this process categories such as Dinamarca (Denmark), Alemania (Germany), República Checa (Czech Republic) are retrieved. The members of these categories contain the desired wikipages of the countries. However, the categories of countries are very noisy because they contain several other types of Wikipages besides the Wikipage of the country. For example the Category Austria contains wikipages such as *Servicio Austriaco de la paz* (Austrian Peace Service*), Fuerza aérea austriaca* (Austrian Air force) and so on, which are not of the type the user is requiring.

The query is also used to perform a text search on the Yago and DBpedia class indexes. For the query mentioned we retrieved the type *country* (for both the Yago classes and DBpedia ontology). With this information we can construct the set defining the target NE type: country-Yago/DBpedia, Category *Paises de Europa (*European Countries), Category *Dinamarca* (Denmark), Category *Alemania* (Germany), etc.

We construct the first candidate set by searching the Wikipage index, specifying the categories found and returning only the pages that contains in the Yago or DBpedia class layer the type país (country). The result set contains the desired Wikipages of the European countries. Pages like *Fuerza aérea austriaca* (Austrian Air force) are filtered because their types don't match the Yago/DBpedia class country.

Then, we construct a query using the second noun phrase *el bidet* (the bidet), which lead to the query with the same words since no matches are found in WordNet, the redirect link dictionary and the DBpedia ontology in the query expansion phase.

We retrieve the pages and categories associated to Bidet. In this process we obtain the Wikipage of the same name. Then this Wikipage is extended to obtain the entities pointed to and that point to the Wikipage Bidet. In this extension we obtain Wikipages such as Porcelain, Asia, Europe, Storm drain, Greece, Italy, Portugal, Argentina, Uruguay, among others. The wikipages obtained are filtered according the target NE type, filtering the pages that don't correspond to the Yago/DBpedia types (the category filtering is performed by the set intersection). The resulting wikipages form the second candidate entity set

Then, the two sets are intersected. The intersection is the output of the system. In this case the output is: *Grecia*(Greece), *Italia* (Italy), *España* (Spain) and *Portugal* (Portugal).

Chapter 4

Experimentation and Results

In this chapter we present the results obtained by the system using the 50 topics released in the GikiCLEF 2009 track and we analyze the performance of the key components of the system. GikiCLEF is an evaluation task under the scope of CLEF which evaluates systems that address complex geographical topics on the Wikipedia collection. These topics are answered by a set of Wikipedia document titles.

The topics were designed to represent realist user needs of different cultural backgrounds. The topics and the data collections are given in 10 languages (English, Bulgarian, German, Norwegian Bokmål, Norwegian Nynorsk, Italian, German, Dutch, Romanian, and Portuguese). The multi-lingual nature of the task is also reflected in the cultural bias shown by some topic, in which correct answers can only be found in a particular language version of the collections.

For example the topic *Which countries have Italian as an official language?* is answered correctly by listing wikipages such as *Italy*, *San Marino* and *Switzerland*. Countries with important Italian minorities or large communities of Italian speakers such as *Argentina* and *Uruguay* aren't considered correct answers because Italian is not established as an official language. Similarly, regions in which Italian language has special or official status such as *Santa Tereza (Brazil)* and *Vila Velha (Brazil)* aren't considered correct answers because the type of these answers (regions) don't match the one asked in the topic (countries). Note that it is more likely to find answers to this question in the Italian Wikipedia collection than in the other collections.

The GikiCLEF official topic list was released the 15 of May, 2009 and the submission deadline was set to the 31 of May. The complete list of topics can be found in Appendix A. The official results were made available the 17 of June. The evaluation was carried out manually by several assessors to increase the quality of the evaluation and resolve conflicts.

Unfortunately by the submission dates the system was still on its middle development stages, for this reason it was only possible to submit a result list with few of the features mentioned in the previous chapter, which lead to poor results. Nonetheless, we collected all the results submitted by the 17 participant to build a list with the correct answers for the 50 topics included in this track edition. We utilized this list to evaluate the performance of our final system and evaluate the impact of the features and modules described in the previous chapter.

We assume that the total numbers of correct documents in the collection for each topic are the ones given in the list. Although there are other correct answers in the collection, we consider that this evaluation methodology provides a fair testbed to compare our system with the other participants and study the performance of the components of our system.

The list contains a total of 1013 correct answers for all the languages available. We found 105 correct answers in Spanish, which is the language of the collection processed in the work presented in this thesis. Figures 6 and 7 show the distribution on answers for all the languages and Spanish respectively. Note that several questions weren't answered by all the participants (6 questions considering all the languages and

18 for Spanish) and the average number of correct answers for the Spanish collection is 2.1 per topic which provide evidence of the high difficulty of the task.

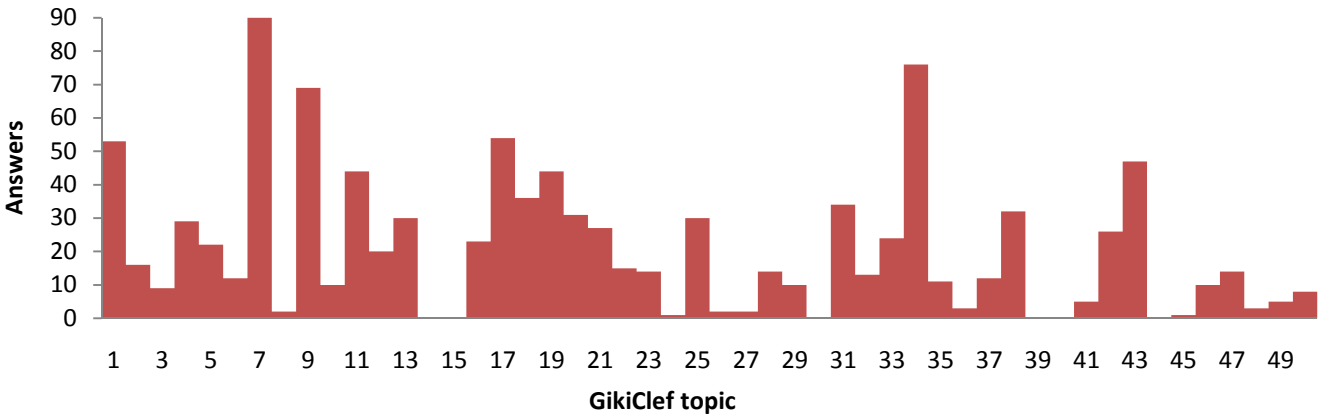


Figure 6. Correct answers distribution for all the languages

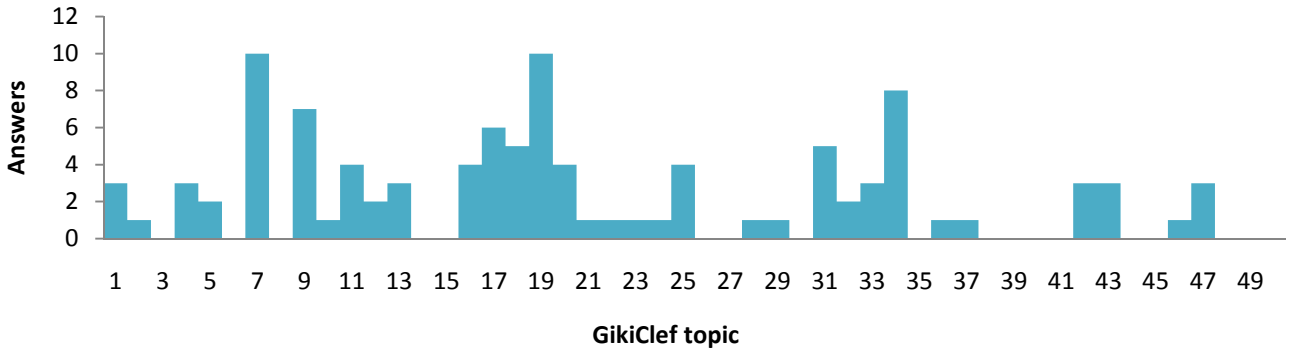


Figure 7. Correct answers distribution for Spanish

We carried out two experiments. In the first experiment we compare the performance gain obtained with the most relevant features of the system: classification and filtering of entities using the Yago-DBpedia types, query expansion and Wikipedia category expansion (inclusion of subcategories).

In the second experiment we evaluated the performance gain obtained by the query expansion module considering that we are utilizing three approaches: WordNet synonyms expansion, ontological expansion using the DBpedia ontology and the expansion provided by the redirect links of Wikipedia, which are also used as an attempt to perform nominalization.

In the final part of this chapter we provide a comparison of the results obtained by our system and the participants of GikiCLEF and a discussion of these results.

4.1 Gain performance of the features of the system

Figure 8 and 9 show that the three features evaluated (Yago-DBpedia type mapping and filtering, query expansion, category expansion) increases the precision, recall and the GikiCLEF score of the system. Additionally, it is shown that the system obtains its best performance when all these features are combined. Recall that the GikiCLEF score is calculated using Equation 1. This equation rewards systems that retrieve answers in several languages. Given that the score is calculated using the sum of correct answers for all the topics, collection of languages that don't contain information on a particular topic are not penalized. This equation also stress harder the difference of the systems based on the number of correct answers retrieved which is convenient in this task since the average of correct answers per topic is small.

$$score \quad = c_{en}\frac{c_{en}}{n_{en}} + c_{nl}\frac{c_{nl}}{n_{nl}} + c_{es}\frac{c_{es}}{n_{es}} + \cdots,$$

Equation 1. Score function utilized in GikiCLEF. c_x corresponds to the number of correct answers in language *x*. n_x refers to the total number of answers in language *x*.

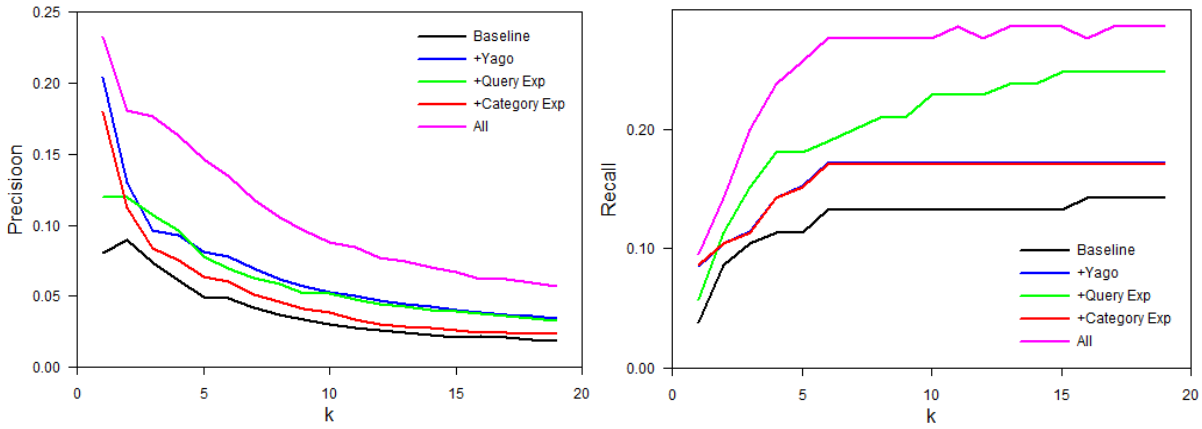
The Query Expansion module significantly increased the number of hits of the system (as can be seen from the coverage graph). This occurs because it is possible to map Wikipedia categories that are not matched with this extension. For example in the topic *Escritores alemanes que sean ciudadanos honorarios de Suiza (*German writers who are Honorary Citizens in Switzerland) the word *alemanes* (germans) is extended using the Wikipedia redirect links to obtain its nominal form, leading to phrase *escritores alemanes Alemania* (writers germans Germany). This makes it possible to map the Wikipedia category Escritores de Alemania (Writers from Germany) which contains the correct answer *Hermann Hesse*.

The topic *Nombre poetas rumanos que hayan publicado volúmenes de romances antes de 1941* (Name Romanian poets who published volumes with ballads until 1941) illustrates the gain obtained using the synonym expansion. In this topic the word poetas (poets) is extended with the synonyms lyricists, wordsmith, novelist, writer, among others. Along with the nominalization of the word *rumanos* to *Rumania* (Romanians- Romania), this extension makes possible the mapping of the category *Poetas de Rumania* (Rumanian poets) which contains the answers *Vasile Alecsandri*, *Ienăchiţă Văcărescu, Tristan Tzara* and *Ion Barbu.*

The Yago-DBpedia mapping particularly increases the precision of the system as it is observed in the Figure 8.A. This results is expected because this mapping is used to filter the candidate entities that don't correspond to the target type, thus cleaner result list are produced. Figure 8.B shows that recall is also increased; nonetheless the gain is lower than the one obtained with the query expansion procedure.

According to Figures 8 and 9, the expansion of Wikipedia categories seems to be the feature with the smallest impact in the performance of the system. However it is important to point out that not always is appropriate to measure the performance gain of each feature independently because the increase of precision and recall is often obtained by the interaction of these functionalities. For example in the topic *Nombre montañas chilenas que tengan nieves perpetuas* (Name Chilean mountains with permanent snow) the query expansion feature is used first to extend the word chilenas (Chileans) by using the Wikipedia redirect links to obtain its nominal form, producing *montañas chilenas Chile* (mountains Chileans Chile*).*

This makes possible to map the Wikipedia category *Montañas de Chile* (Mountains of Chile) where correct answers as *Licancabur* and *Llullaillaco* are found. The category expansion procedure is applied o the category *Montañas de Chile* (Mountains of Chile) to obtain the subcategories*: Volcanes de Chile (*Chilean Volcanos*) and Cerros de Chile (*Chilean Monts*).* The answers *Parinacota* and *Tupungato* are extracted from the category *Volcanes de Chile*



(A) (B)

Figure 8. Precision ad Recall considering different modules of the system. *k* refers to the parameter used to truncate the number of hits returned by the IR engine in the restriction filtering phase. In 7 (B) the +Yago and +Category Exp lines coincide by coincidence

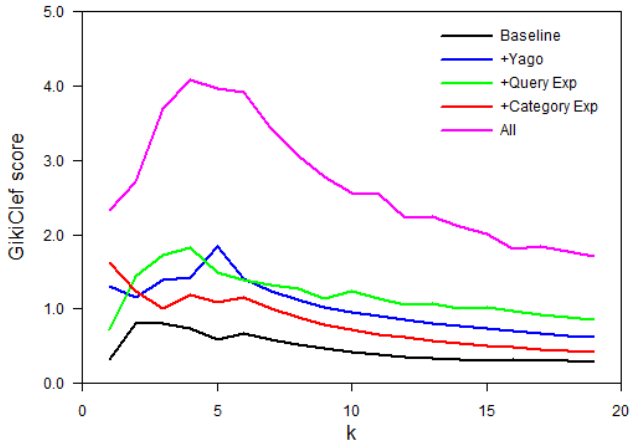
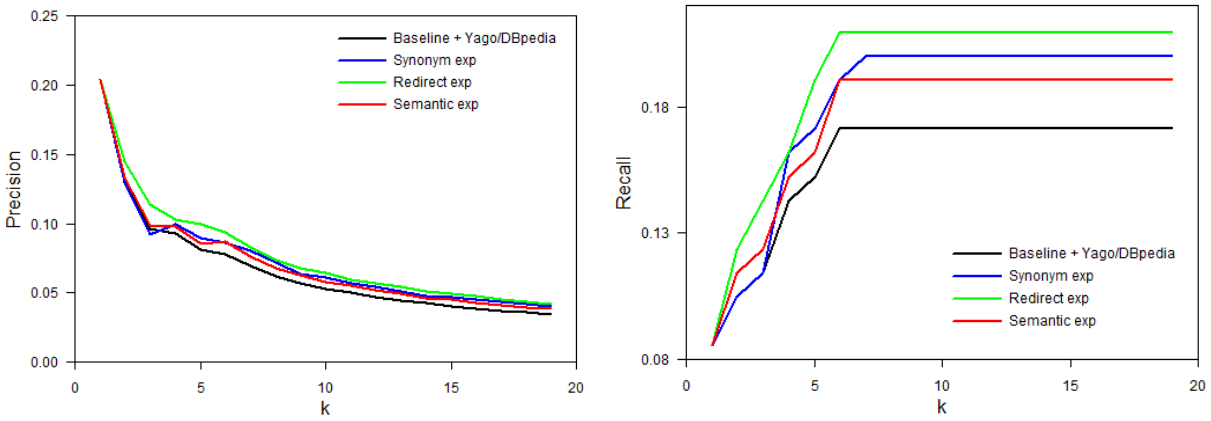


Figure 9. GikiCLEF scores considering different modules of the system. *k* refers to the parameter used to truncate the number of hits returned by the IR engine in the restriction filtering phase.

The maximum GikiCLEF scored obtained by the system is 4.08, this value is obtained using 4 as cut-off parameter. This parameter is used to limit the number of pages included from the output of the IR engine in the restriction filtering. Figure 8 shows that for all the values of this parameter the best performance is obtained when all the features of the system are used together. This figure also shows that the query expansion produces the greatest gain for most of the cut-off values. Similarly the category expansion feature produces the smallest impact in the GikiCLEF score of the system.

4.2 Query Expansion experiments

Figures 10 and 11 show in greater detailed the performance gain obtained with the three strategies utilized in the query expansion procedure. The three plots show that the query expansion based on the Wikipedia redirect links contributes the most to increase the precision, recall and GikiCLEF score of the system.



(A) (B)

Figure 10. Precision ad Recall using the query expansion features. *k* refers to the parameter used to truncate the number of hits returned by the IR engine in the restriction filtering phase.

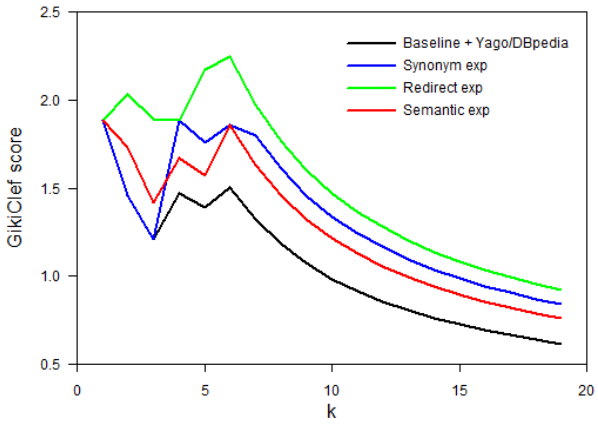


Figure 11. GikiCLEF score obtained using the query expansion features. *k* refers to the parameter used to truncate the number of hits returned by the IR engine in the restriction filtering phase.

This occurs because the redirect links are commonly used to obtain the country that corresponds to demonyms or adjectival forms expressed in the topics, for example *Español* (Spanish) is extended as *España* (Spain). In the Spanish topic list this situation appears in 46% of the topics. The gain of accuracy is obtained because all the Wikipedia categories matched that mention or refer to places don't contain adjectival forms.

The synonyms expansion also provides a boost in the global performance of the system. However the gain is smaller. This may be due to the cross-lingual dictionary which is used to translate the WordNet

synonyms, since first the Spanish word has to be translated to English and then the synsets translated back to Spanish decreasing the coverage of the method.

The semantic expansion had the lowest impact among the query expansion methods because we are using the DBpedia ontology[4] to perform this extension, which contain only 170 entries. We found that this expansion was only effective in the topics 1, 9 and 29. In all these topics the NE type required is *"places"* which is extended using the ontology with the words *city*, *island*, *country*, *area*, *park* and so on.

It is important to mention that these expansion techniques also lead to performance gain when they are used together. For example in the topic 32 *Nombre escritores rumanos que estuvieran viviendo en 2003 en Estados Unidos* (Name Romanian writers who were living in USA in 2003), the redirect links are used to nominalize *Rumanos* (Romanian) to *Rumania* (Romania), and the synonym expansions extends *escritores* (writers) with the word *novelistas* (novelist) and *poetas* (poets). This makes possible to match categories such as *Poetas de Rumania* (Rumanian poets) and *Novelistas de Romania* (Rumanian novelist).

Figure 11 shows the same trend observed in Figure 10. The used of the redirect links leads to the greatest increase of the GikiCLEF score, followed by the synonyms and semantic expansion.

4.3 Comparison of results

In this section we compare the results obtained by our system and the participants of the GikiCLEF 2009 task. Table 5 summarizes the results obtained by the participant systems considering only the Spanish topics.

Participant	Run Score	Answers	Corrects	Precision	Giki score
idornescu	1	82	37	0.4512	16.6951
greasu	1	104	34	0.3269	11.11
raylarson	1	52	16	0.3077	4.923
svenhart	1	3	3	1	3
svenhart	2	72	12	0.1667	2
svenhart	3	62	9	0.1452	1.3065
Talp-upc	1	60	4	0.0667	0.2667
Talp-upc	2	143	2	0.0140	0.0280
UAICGIKI09	1	642	2	0.0031	0.0062
Our system		*153*	*25*	*0.1633*	*4.08*

Table 5. Results obtained by the participants of GikiCLEF 2009 on the Spanish topics

This table shows that our system obtains a decent number of correct answers since only two systems obtained more hits. Nonetheless the total number of answers is high compared to most of the systems. This result suggests that the mapping of Wikipedia categories and yago/dbpedia classes used to define the NE entity type and construct the candidate entity set is not particularly problematic in our system since we almost obtain as many hits as the idornescu and greasu system when the cutoff parameter is increased

[4] http://www4.wiwiss.fu-berlin.de/dbpedia/dev/ontology.htm

to relax the filtering of restrictions (increasing number of wikipages returned by the IR engine in the filtering phase). However, this result shows that the filtering of candidate is more problematic in our system.

To have a better understanding of these results we calculated the GikiCLEF score obtained by the idornescu system and our system for each topic. These results are summarized in Figure 12.

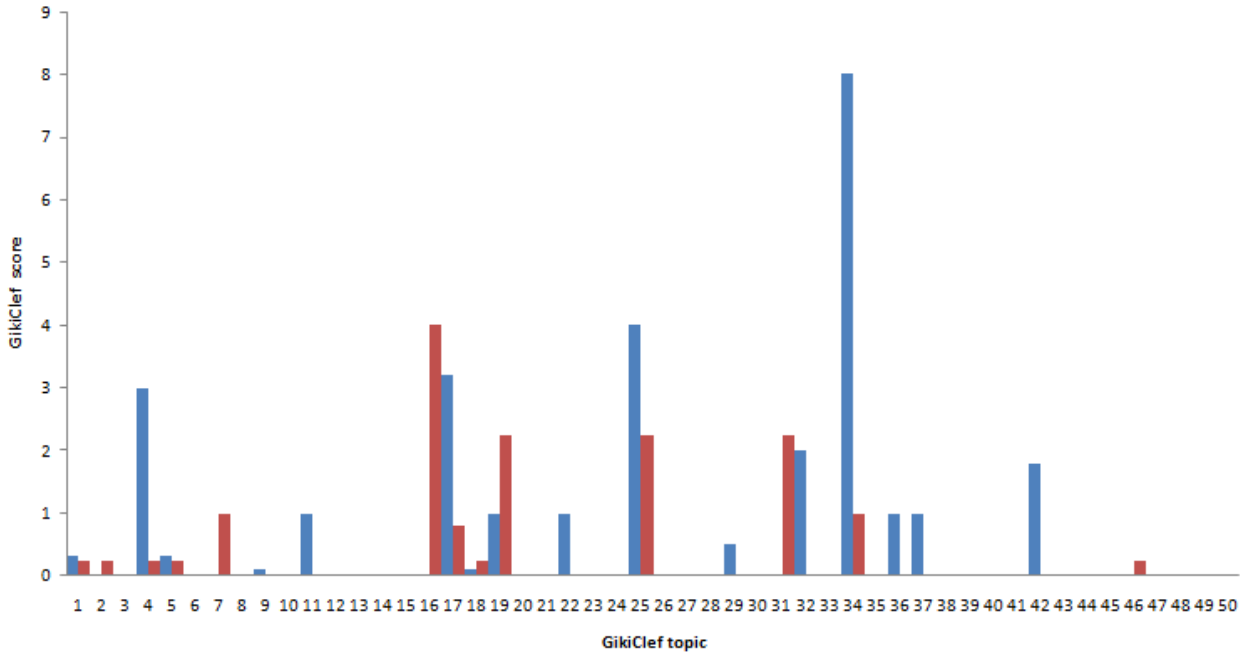


Figure 12. Distribution of the GikiCLEF score obtained by the idornescu system(blue) and our system(red)

We verified the topics in which the idornescu system outperforms our system. It was found that several of the answers that weren't included by our system did not mention in the Wikipedia content the restrictions, thus the text search and matching performed in the filtering phase becomes ineffectual. However, the restrictions of these topics are commonly mentioned in the content of alternative language version of the wikipages (mainly in the English version). This suggests that these pages were found by analyzing a Wikipedia collection in a different language and then added to the Spanish result list by using the cross-lingual links. In fact we found that most of the systems submitted to GikiCLEF reported the greatest amount of answers in the English Wikipedia collection. This situation is expected since this collection is the biggest and better structured one (greater amount of content, linking system, templates and categories.)

Chapter 5

Conclusions

5.1 Conclusions

This thesis set out to propose a system to retrieve entities according the NE type and characteristics specified in a topic or question given in natural language, thus the task addressed by our system is analogous to the one defined in the GikiCLEF track. Although the system we proposed was fed with the Spanish Wikipedia collection, the same system architecture can easily be used in other languages by replacing few of the components of the system modules, concretely: the POS tagger, the NER and the Wikipedia semi-structured resources (categories, cross-lingual, redirect link dictionaries). Recall that the cross-lingual dictionary is used to map the DBpedia and Yago semantic resources from English into Spanish.

We showed that our system obtained a GikiCLEF score of 4.08, 0.168 precision and 0.238 recall. These results are highly promising given that the system would rank in the fourth position among the 17 participants of the task and because we only processed the Spanish Wikipedia collection, which is significantly less developed and completed than the English collection. We found that most of the other participants employed the content of the wikipages in the English Wikipedia, particularly those who obtained better ranking. Thus the results suggest that our system provides an appropriate and effective architecture to perform entity retrieval on Wikipedia, which was the main goal of this thesis. Additionally it provides an organized framework to perform further research in each one of the modules of the system: question analysis, query expansion, generation of candidate entities and entity filtering.

We proposed three novel methods to perform query expansion in the problem of entity retrieval. The first method consists of exploiting the redirect links of Wikipedia by building a dictionary that specifies alternative names to refer to some of the entities in the collection. We found that this dictionary is a valuable source of information to normalize entity names and nominalize adjectival forms. This method provided the greatest performance gain among the three methods presented. The second query expansion method consists of using WordNet and the cross-lingual links to translate English to Spanish. Although the coverage of this method is limited by the size of the cross-lingual dictionary, we found that this method also increases the accuracy of the system. The final method proposed was the semantic expansion which consists of extending the words in the query by using the DBpedia ontology. Although the performance gain obtained with this method was smaller than the other two, we consider that the use of larger ontologies would highly benefit the system since the coverage of the method is limited by the size of the ontologies.

We also introduced a novel method to employ the Yago and DBpedia semantic resources to map the named entity type that the user requires and filter the candidate entity set according this type; this method is used to improve previous approaches in which the type definition and generation of the candidate entity set is based on the Wikipedia categories. We showed that the introduction of Yago and DBpedia in these tasks improved the results of the baseline system.

In order to use the Yago and DBpedia resources we translated the Yago types and DBpedia ontology using the dictionary built from the cross-lingual links. However, the results were revised and refined

manually to guarantee the quality of the translation. These results represent another contribution of this thesis since other information retrieval systems of Spanish data could make use of this data.

We also showed that the methods to filter the entities that don't satisfy the restrictions imposed in the query are effective. The first approach to model these restrictions is to map Wikipedia categories and Wikipages associated to the noun phrases of the input question mentioning these constrains. These items are expanded using the link structure of Wikipedia. Then, the resulting set is filtered using the Yago/DBpedia class and the Wikipedia Categories that represent the target NE type (by using set intersection). This method assumes that all the Wikipages of the correct type which are pointed from or point to the Wikipages mapping the restrictions are correct answers. For example in the topic *Name places where Goethe fell in love,* it is assumed that all the Wikipages of places that are pointed from or point to the Wikipage *Goethe* correspond to correct answers. The results obtained show that this assumption is not as strong as it seems and that this method indeed leads to correct answers frequently.

The other method to filter the wikipages is to perform a text search of the restriction in the candidate entity set by using the IR engine. The text search is performed on the content of the wikipages (infoboxes values are also included in the content). We showed that the score assigned by the IR engine is an appropriate suitable value to rank the entities in the candidate set, allowing the selection of the top *N* ranked pages as the most likely correct answers. Note that this type of entity scoring is also used in [**20**], where the task to solve is to retrieve and rank entities on the Wikipedia collection using a Wikipedia category and entities exemplifying the correct answers. We found it unsuitable to use the other ranking measures proposed in [**20**] because these measures are based on the entities and categories given as examples, which is information that is not provided in the task we are solving.

Last but not least, we showed that the methods proposed to extract the NE type and restrictions from the user's questions (Algorithm 2 and 3) correctly identified the noun phrases containing this information

5.2 Future work

Although satisfactory results were obtained with the current architecture of the system and the methods proposed in each one of these modules, there is still place for improvement in several parts of the system.

The query analysis module can be extended to study blind relevance feedback techniques as an attempt to increase the recall of the system. Similarly the addition of larger ontologies and geographical resources (such as gazetteers) may lead to an increase in the performance of the system.

The filtering of restrictions can be improved by extending the use of the semantic resources by studying methods to use the Yago and DBpedia knowledge to perform a deeper evaluation of the restrictions.

The system can be further enhanced by adding support to other languages, particularly for the English collection since it is the most completed Wikipedia. However it is also encouraged to add support to heterogeneous languages to increase the coverage of cultural dependent topics. Additionally, since the English Wikipedia is the most developed collection, it is desirable to automatically complete the semi-structured data of the Wikipedia collections in other languages based on the English version. Previous approaches have already been applied to perform automatic completion of Wikipedia templates [**28**]. Our system would be greatly beneficiated by studies that perform this kind of completion for the Wikipedia resources utilized such as Wikipedia categories, link structure, redirect links and so on.

References

[1] E. Voorhees, "The trec-8 question answering track report.," *Text Retrieval Conference*, 1999.

[2] S. Schlobach, A. D, M. De Rijke, and V. Jijkoun, "Data-driven Type Checking in Open Domain Question Answering," *J. of Applied Logic*, p. 5(1):121–143, 2007.

[3] G. Bouma, et al., "Linguistic knowledge and question answering," *TAL. Volume 46*, pp. 15-39, 2005.

[4] U. Hermjakob, E. Hovy, and C. Lin, "Automated Question Answering in Webclopedia - A Demonstration," *In Proceedings of ACL-02*, 2002.

[5] K. Balog and M. De Rijke, "Finding Experts and their Details in E-mail Corpora," *In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors*, p. WWW,ages1035–1036ACM, 2006.

[6] K. S. Tjong, "Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition," *In: Proceedings of CoNLL-2002, Taipei, Taiwan*, pp. 155-158, 2002.

[7] H. Rode, *From Document to Entity Retrieval, phD Dissertion*. University of Twente, 2008.

[8] I. Soboroff, N. Craswell, and A. De Vries, "Overview of the TREC 2005 Enterprise Track.," in *Proceedings of the Fourteenth Text Retrieval Conference, TREC 2005*, Gaithersburg, Maryland, 2005.

[9] D. Santos, et al., "Getting geographical answers from Wikipedia: the GikiP pilot at CLEF," *Cross Language Evaluation Forum: Working Notes for the CLEF 2008 Workshop*, 2008.

[10] O. Medelyan, C. Legg, D. Milne, and I. Witten, "Mining meaning from Wikipedia," *University of Waikato, New Zealand*, 2008.

[11] J. Leveling and S. Hartrumpf, "Inferring location names for geographic information retrieval. ," in *In Advances in Multilingual and Multimodal Information Retrieval: 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007*, vol. 5152, 2008, p. 773–780.

[12] S. Hartrumpf, "Question answering using sentence parsing and semantic network matching," *In Multilingual Information Access forText, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum, CLEF 2004*, vol. 3491, pp. 512-521, 2005.

[13] H. Helbig, "Word class functions for syntactic-semantic analysis," *In Preceedings of the 2nd International Conference on Recent Advances in Natural Language Processing*, pp. 312-317, 1997.

[14] S. Hartrumpf, H. Helbig, and R. Osswald, "The Semantically Based Computer Lexicon HaGenLex -

Structure and Technological Environment," *Traitement automatique des langues*, pp. 81-105, 2008.

[15] N. Cardoso, "REMBRANDT - Reconhecimento de Entidades Mencionadas Baseado em Relacoes e Analise Detalhada do Texto," *Desafios na avaliacao conjunta do reconhecimento de entidades mencionadas: Actas do Encontro do Segundo HAREM*, 2008.

[16] D. Santos, N. Seco, C. Nuno, and R. Vilela, "HAREM: An Advanced NER Evaluation Contest for Portuguese," *Proceedings of LREC*, pp. 1986-1991, 2006.

[17] P. Boldi and S. Vigna, "MG4J at TREC," in *In The Fourteenth Text Retrieval Conference (TREC 2005) Proceedings*, 2005.

[18] T. J. Tapanainen et al, "A non-projective dependency parser," *In Proceedings of the 5th Conference of Applied Natural Language Processing*, pp. 64-71, 1997.

[19] O. Gospodnetic and E. Hatcher, *Lucene in Action (In Action series)*. USA: Manning, Greenwich, CT, 2004.

[20] A. Vercoustre, J. Thom, and J. Pehcevski, "Entity Ranking in Wikipedia," *Proceedings of the 2008 ACM symposium on Applied computing*, 2008.

[21] F. Suchanek, K. Gjergji, and G. Weikum, "Yago: a core of semantic knowledge," *Preceeding of the 16th international conference on Word Wide Web*, pp. 697-706, 2007.

[22] S. Auer, et al., "DBpedia: A Nucleus for a Web of Open Data," *6th International Semantic Web Conference (ISWC 2007)*, 2007. [Online]. http://en.wikipedia.org/wiki/DBpedia

[23] C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[24] C. Baueret and G. King, *Java Persistence with Hibernate*. Manning, 2007.

[25] P. Willett, "The Porter stemming algorithm: then and now," *Program: electronic library and information systems*, vol. 40, no. 219-223, 2006.

[26] X. Carreras. (2002) Resources on Named Entity Recognition and Classification. [Online]. http://www.lsi.upc.es/~nlp/tools/nerc/nerc.html

[27] J. Finkel, T. Grenager, and M. Christopher, "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling," *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*, 2005.

[28] G. Bouma, S. Duarte, and Z. Islam, "Cross-lingual Alignment and Completion of Wikipedia

Templates," in *Proceedings of the Third International Workshop on Cross Lingual Information access, Addressing the Information Need of Multilingual Societies (CLIAWS3)*, 2009.

Appendix A

GikiCLEF 2009 topics in Spanish

GC-2009-01: Nombre los lugares de Italia que haya visitado Ernest Hemingway a lo largo de su vida.
GC-2009-02: ¿Qué países tienen los colores blanco, verde y rojo en su bandera nacional?
GC-2009-03: ¿En qué países afuera de Bulgaria se han publicado opiniones sobre las ideas de Petar Dunov?
GC-2009-04: Nombre poetas rumanos que hayan publicado volúmenes de romances antes de 1941.
GC-2009-05: ¿Qué obras literarias de escritores no rumanos tienen por tema los Cárpatos?
GC-2009-06: ¿Qué violinistas holandeses ocuparon el puesto de concertino en la Orquesta Real del Concertgebouw durante el siglo XX?
GC-2009-07: ¿Qué capitales de provincias holandesas recibieron sus derechos de ciudad antes del siglo XIV?
GC-2009-08: ¿Qué autores que nacieron en el bosque de Bohemia han escrito sobre él?
GC-2009-09: Nombre lugares donde Goethe se haya enamorado.
GC-2009-10: ¿Qué ciudades flamencas tenían en 2008 un restaurante con dos o tres estrellas Michelín?
GC-2009-11: ¿Qué belgas han ganado el Tour de Flandes exactamente dos veces?
GC-2009-12: Monarquías europeas cuyo jefe de estado sea actualmente una mujer.
GC-2009-13: Novelistas europeos románticos y realistas del siglo XIX que murieran de tuberculosis.
GC-2009-14: Nombre enfermedades raras que tengan dedicadas centros de investigación en Europa.
GC-2009-15: Liste los ingredientes principales de la Cassata.
GC-2009-16: ¿En qué países europeos se suele usar el bidé?
GC-2009-17: Nombre las cinco regiones italianas que tengan un estatuto especial.
GC-2009-18: ¿En qué provincias de la Toscana se produce el Chianti?
GC-2009-19: Nombre montañas chilenas que tengan nieves perpetuas.
GC-2009-20: Liste los nombres de las secciones noroccidentales de los Alpes.
GC-2009-21: Nombre los afluentes de la margen izquierda del río Po.
GC-2009-22: ¿Qué selecciones sudamericanas de fútbol visten de color amarillo?
GC-2009-23: Nombre museos americanos que tengan alguna obra de Picasso.
GC-2009-24: ¿Qué países han ganado un campeonato de Europa de fútbol sala celebrado en España?
GC-2009-25: Nombre pilotos españoles que hayan corrido en Minardi.
GC-2009-26: ¿Qué luchadores búlgaros han sido galardonados con el cinturón de diamantes?
GC-2009-27: ¿Qué bandas holandesas deben su nombre al de un futbolista búlgaro?
GC-2009-28: Nombre estados costeros que tengan refinerías de Petrobras.
GC-2009-29: poblaciones sobre el círculo polar ártico con más de 100.000 habitantes.
GC-2009-30: ¿Qué fabricantes japoneses de automóviles tienen cadenas de producción o de montaje en Europa?
GC-2009-31: ¿Qué países tienen el italiano como lengua oficial?
GC-2009-32: Nombre escritores rumanos que estuvieran viviendo en 2003 en Estados Unidos.
GC-2009-33: ¿Qué países de la Unión Europea tienen parques nacionales en los Alpes?
GC-2009-34: ¿Qué ochomiles pertenecen al menos parcialmente a Nepal?
GC-2009-35: ¿Qué montañas rumanas están declaradas reserva de la biosfera?
GC-2009-36: Nombre cuevas rumanas donde se hayan encontrado restos fósiles humanos del Paleolítico.
GC-2009-37: ¿Qué músicos noruegos fueron encarcelados por quemar iglesias?
GC-2009-38: ¿Qué cataratas noruegas tienen una altura superior a 200 metros?
GC-2009-39: jugadores de fútbol de equipos nacionales escandinavos que tengan hijos que hayan jugado en equipos ingleses.
GC-2009-40: ¿Qué ríos de Renania del Norte-Westfalia tienen aproximadamente 10 km de longitud?
GC-2009-41: Cocineros nacidos en Austria que hayan recibido una estrella Michelin.

GC-2009-42: Partidos políticos del consejo nacional austriaco que hayan sido fundados después del fin de la segunda guerra mundial.
GC-2009-43: Estaciones de esquí austriacas cuya longitud total de pistas sea de al menos 100 km.
GC-2009-44: Encuentre variedades de uva austriaca cuya área de viñedos sea menor a 100 hectáreas.
GC-2009-45: Encuentre ganadores de programas suizos de casting.
GC-2009-46: Escritores alemanes que sean ciudadanos honorarios de Suiza.
GC-2009-47: ¿Qué ciudades alemanas tienen más de una universidad?
GC-2009-48: ¿Qué películas de habla alemana han recibido una nominación a los Óscar?
GC-2009-49: Pilotos de Fórmula 1 que se hayan mudado a Suiza.
GC-2009-50: ¿Qué suizos ganaron en snowboard medallas olímpicas durante los Juegos Olímpicos de Invierno de 2006?

Appendix B

GikiCLEF 2009 topics in English

GC-2009-01: List the Italian places where Ernest Hemingway visited during his life.
GC-2009-02: Which countries have the white, green and red colors in their national flag?
GC-2009-03: In which countries outside Bulgaria are there published opinions on Petar Dunov's (Beinsa Duno's) ideas?
GC-2009-04: Name Romanian poets who published volumes with ballads until 1941.
GC-2009-05: Which written fictional works of non-Romanian authors have as subject the Carpathians mountains?
GC-2009-06: Which Dutch violinists held the post of concertmaster at the Royal Concertgebouw Orchestra in the twentieth century?
GC-2009-07: What capitals of Dutch provinces received their town privileges before the fourteenth century?
GC-2009-08: Which authors were born in and write about the Bohemian Forest?
GC-2009-09: Name places where Goethe fell in love.
GC-2009-10: Which Flemish towns hosted a restaurant with two or three Michelin stars in 2008?
GC-2009-11: What Belgians won the Ronde van Vlaanderen exactly twice?
GC-2009-12: Present monarchies in Europe headed by a woman.
GC-2009-13: Romantic and realist European novelists of the XIXth century who died of tuberculosis.
GC-2009-14: Name rare diseases with dedicated research centers in Europe.
GC-2009-15: List the basic elements of the cassata.
GC-2009-16: In which European countries is the bidet commonly used?
GC-2009-17: List the 5 Italian regions with a special statute.
GC-2009-18: In which Tuscan provinces is Chianti produced?
GC-2009-19: Name mountains in Chile with permanent snow.
GC-2009-20: List the name of the sections of the North-Western Alps.
GC-2009-21: List the left side tributaries of the Po river.
GC-2009-22: Which South American national football teams use the yellow color?
GC-2009-23: Name American museums which have any Picasso painting.
GC-2009-24: Which countries have won a futsal European championship played in Spain?
GC-2009-25: Name Spanish drivers who have driven in Minardi.
GC-2009-26: Which Bulgarian fighters were awarded the "Diamond belt"?
GC-2009-27: Which Dutch bands are named after a Bulgarian footballer?
GC-2009-28: Find coastal states with Petrobras refineries.
GC-2009-29: Places above the Arctic circle with a population larger than 100,000 people
GC-2009-30: Which Japanese automakers companies have manufacturing or assembling factories in Europe?
GC-2009-31: Which countries have Italian as an official language?
GC-2009-32: Name Romanian writers who were living in USA in 2003.
GC-2009-33: What European Union countries have national parks in the Alps?
GC-2009-34: What eight-thousanders are at least partially in Nepal?
GC-2009-35: Which Romanian mountains are declared biosphere reserves?
GC-2009-36: Name Romanian caves where Paleolithic human fossil remains were found.
GC-2009-37: Which Norwegian musicians were convicted for burning churches?
GC-2009-38: Which Norwegian waterfalls are higher than 200m?
GC-2009-39: National team football players from Scandinavia with sons who have played for English clubs.
GC-2009-40: Which rivers in North Rhine Westphalia are approximately 10km long?

GC-2009-41: Chefs born in Austria who have received a Michelin Star.
GC-2009-42: Political parties in the National Council of Austria which were founded after the end of World War II
GC-2009-43: Austrian ski resorts with a total ski trail length of at least 100 km
GC-2009-44: Find Austrian grape varieties with a vineyard area below 100 ha.
GC-2009-45: Find Swiss casting show winners.
GC-2009-46: German writers who are Honorary Citizens in Switzerland.
GC-2009-47: Which cities in Germany have more than one university?
GC-2009-48: Which German-speaking movies have been nominated for an Oscar?
GC-2009-49: Formula One drivers who moved to Switzerland.
GC-2009-50: Which Swiss people were Olympic medalists in snowboarding at the Winter Olympic Games in 2006?

Appendix C

Answers obtained in GikiCLEF 2009

Topic	Answers
GC-2009-01	Milán
GC-2009-02	Gales
GC-2009-04	Iancu_Văcărescu_2253
GC-2009-05	Drácula
GC-2009-07	Zwolle,Arnhem
GC-2009-16	España,Andorra,Grecia,Italia
GC-2009-17	Lombardía,Sicilia
GC-2009-18	Provincia_de_Arezzo_9bce
GC-2009-19	Licancabur,Parinacota__(volcán_),Llullaillaco
GC-2009-25	Adrián_Campos_f93d,Fernando_Alonso_f1dc,Marc_Gené_a09f
GC-2009-31	Ciudad_del_Vaticano_16a8,Italia,San_Marino_54a0
GC-2009-34	Mount_Everest_bc8a,Kanchenjunga
GC-2009-46	Hermann_Hesse_69e1