# Open-Domain Web-Based List Question Answering with LX-ListQuestion

Patricia Nunes Gonçalves, António Branco
University of Lisbon
Edifício C6, Departamento de Informática
Faculdade de Ciências, Universidade de Lisboa
Campo Grande, 1749-016 Lisboa
{patricia.nunes, antonio.branco}@di.fc.ul.pt

## ABSTRACT

Research in the field of Question Answering (QA) has experienced many advances. However, the development of QA systems continues to be a challenging task. We present a QA Web Application, LX-ListQuestion that focuses on answering list questions where the answers are extracted and composed from several documents retrieved from the Web. The system exploits the redundancy of information available in the Web, combined with word occurrence patterns to improve QA accuracy. Answers are presented in the form of a Word Cloud that uses font size to depict relevance. This paper addresses the main problems that must be dealt with when answering list questions and describes the architecture of the system. We also present an experimental evaluation using a set of questions from QA competition.

## Categories and Subject Descriptors

D.2 [**Software Engineering**]: General; H.3.5 [ **Information Systems**]: Online Information Services

## General Terms

Question Answering

## Keywords

List Question, Web as Corpus, Web System

## 1. INTRODUCTION

With the growth of the Internet, more people are searching for information on the Web. The combination of web growth and improvements in Information Technology has reignited the interest in Question Answering (QA) systems. QA is a type of information retrieval combined with natural language processing techniques that aims at finding answers to natural language questions. QA can be regarded as the next step beyond mere keyword search. In a search engine, the user inserts a few keywords and gets as a result links and snippets. The task of finding the desired answer among the results that were returned then falls on the user. From the point of view of QA, in turn, the users submit a question in natural language and the system searches within the documents for the exact answers.

Answering questions is not an easy task, and questions have various levels of complexity. When considering questions, factual questions immediately come to mind (eg: When did Nelson Mandela die?), however, the QA area has expanded beyond factual questions towards more complex questions. One of the most common types of complex questions is list questions. The list questions are questions for which there is a list of answers, e.g., *In which countries Portuguese is an official language?* List answers: *Angola, Brazil, Cape Verde, Guine Bissau, Mozambique, Portugal, Macau, East Timor and Sao Tome and Principe.*

The basic process of searching answers for factual questions may be the same when the user searches for list answers. However, the level of complexity in list question should be taken into account. Consider this scenario: The user wishes to find a list of European countries. To do this, the user inserts a few keywords into a search engine, for instance, *european countries*, and is quite likely to find a web page containing the desired information among the first hits returned by the search engine. In other words, when the information that is needed is trivial, and a web page with the full answer already exists, a search engine may help with this problem. However, when the information that is needed is non-trivial and it is found spread over several texts, a lot of human effort is required to gather the various separate pieces of data into the desired result, which is not an easy task.

The current state-of-the-art, be it in Information Retrieval or Question Answering, does not provide yet a perfect way to tackle this complex problem.

Users do not generally want to go through several documents and put a lot of effort in finding the desired answer. Ideally, they would prefer to quickly get a precise answer and go on to make use of it instead of spending time searching and compiling the answer from pieces spread over several documents. Our purpose is to provide better QA solutions to users, who desire direct answers to their queries, using approaches that deal with the complex problem of extracting answers found spread over several documents and use


*WIMS* '14, June 2-4, 2014, Thessaloniki, Greece

http://dx.doi.org/10.1145/2611040.2611066.

them to compile a list of answers that are the most accurate possible. The development of LX-ListQuestion takes advantage of the fact that, when doing QA over free text captured dynamically in the Web, the answers may appear redundantly in many places and in many forms. Our approach to address the problem of answering list questions is to explore this redundancy. To build on this redundancy, we use techniques that will be explained in section 4.

**Paper outline:** Section 2 provides an overview of related work with focus on QA System and List Questions. Section 3 introduces List Questions and some examples are presented. The LX-ListQuestion System architecture and experimental evaluation are described on Section 4. Finally, Section 5 concludes with some final remarks.

## 2. RELATED WORK

An overview of QA research can be found in [11]. This section covers related work focusing on List questions. The most common approach is to take a QA system for factoid questions and apply it repeatedly to obtain different answers. Some systems using this approach are [6] and [13] and the performance of these systems is very low, with less than 0.10 f-score value. Other systems explore NLP tools and linguistic resources, mainly named entity recognition, PropBank, NomBank, FrameNet, semantic dependencing, correference resolution, WordNet and ontologies. For instance, [8] reports 0.148 of f-score and [15] reports 0.31 f-score for List questions. The time required for processing is very high and the performance of these systems depend on the performance of the supporting NLP tools.

Other approaches resort to statistical and machine learning methods. The system developed in [12] is based on a statistical model to answer List Questions. The best result is 0.035 f-score in List questions. Machine learning has also been used in the context of Question Answering. The system developed by [14] employs classification techniques to improve the system to find complete and distinct answers. They improve the results of 0.319 to 0.464 f-score with this technique. The system proposed by [10] answers List questions using a clustering machine learning method to group candidate answers that co-occur more often in the collection and achieves 0.287 of recall.

Other systems take advantage from semantic content to answer List questions, e.g. [2], [7], [4]. These achieved competitive results although all information should be stored in a database. This approach seems suitable to QA system that focus on a specific domain where the information source can be limited and more easily stored, but can hardly cope with open-domain QA. The best results of [7] is 0.14 of precision, [4] achieved 0.32 of recall and [2] does not report results.

## 3. LIST QUESTIONS

List questions have been widely studied in the QA domain. For List questions, a system is expected to return not a single answer but a list of answers. In the context of QA research, list questions may appear in three basic forms: (1) a question starting with an interrogative pronoun, (2) a request using an imperative verb and (3) other forms: without interrogative pronoun or imperative verb (usually a complex noun phrase). Table 1 shows some examples of these various forms of List questions.

**Table 1: Examples of List questions**

| Type of List Question | Example |
|---|---|
| Interrogative Pronoum | What European Union countries have national parks in the Alps? |
| Imperative Form | Name rare diseases with dedicated research centers in Europe. |
| Other | Chefs born in Austria who have received a Michelin Star. |

List questions are already complex by themselves but they can also be made even more complex by adding constraints. The most common constraints are: Temporal, Geographic and Quantitative. Temporal constraints are related to time (months, years, centuries, etc.), e.g. *What are the Brazilian poets who published volumes with ballads* ***until 1941****?* Geographic constraints are related to localization (cities, region, countries, continents and so on), e.g. *What are the rare diseases with dedicated research centers* ***in Europe****?* Quantitative constraints are related with how often something happen, e.g. *What Belgians won the Ronde Van Vlaanderen* ***exactly twice****?*

The answers (for a list question) may appear in many places and in many forms. They can be in the same document; when the answer is already a list, e.g., *list of cities in Portugal: Lisbon, Coimbra, Porto and Faro*; or the answers can be spread over multiple documents; e.g., (document A): *Lisbon is the capital of Portugal.* (document B) *Porto is a very important city in Portugal.* In the latter case, a QA system aimed to answering List questions has to find all the answers spread over the several texts and compose the final list of answers.

Our system focuses on answering List questions where the answers are extracted from several documents from the Web.

## 4. LX-LISTQUESTION ARCHITECTURE

The LX-ListQuestion System seeks to answer List questions through the use of Question Answering techniques running over the Web of Portuguese pages, while ensuring that the final answer List is as correct and complete as possible. The system exploits the redundancy of information and combines with word occurrence rules to find correct answers. The system architecture has three main modules: Question Processing, Passage Retrieval and Answer Extraction. Figure 1 shows its architecture and the main modules of the system is described in detail in the following sub-sections.

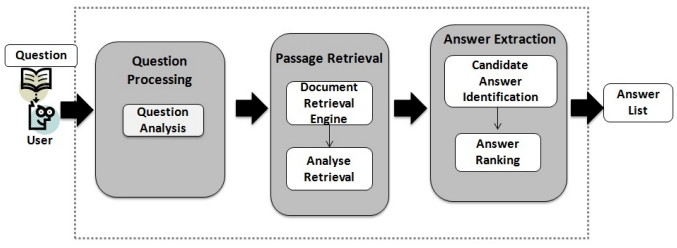


**Figure 1: Question Answering System Architecture.**

## 4.1 Question Processing Module

The Question Processing module is responsible for converting a natural language question into a form that a computer is capable of handling and extracting the information that will be used by subsequent modules. Question Analysis task is responsible for cleaning the questions, i.e. removing question marks, interrogative pronouns and imperative verbs. The system annotates each word with their part-of-speech tag and extracts the following elements of the question: main verb, question target and named entities. This set of extracted words of the question we term as root question. Note that the root question is composed only by proper nouns, common nouns and verbs. The number of words in the root question will settle the strategy that will guide the kernel of the system. The strategy can follow three diferent levels based on the number of words in the root question. Table 2 shows the number of words related with the strategy level. The strategy level leads to how many files of relevant information the system uses to gather the sentences and how many auxiliary lists the system will work with.

**Table 2: Strategy level**

| Root-question Example | #words | Strategy level |
|---|---|---|
| Churches Macau | <3 | 1st |
| National parks Mozambique | 3 | 2nd |
| Typical dishes cuisine Cape Verde | >3 | 3th |

Another task performed by the Question Processing module is the expansion of keywords. This task is made using two different algorithms: Nominal Expansion and Verbal Expansion. With the Nominal Expansion, the system identifies the synonyms and hypernyms of the common nouns. With the Verbal Expansion, the main verb is conjugated into other verbal forms. The expansion of keywords will be used to select relevant sentences from the source text. Table 3 shows an example of this step.

**Table 3: Results of Question Processing and Passage Retrieval Module**

| | |
|---|---|
| Original Question | List Tuscany provinces that produce Chianti. |
| Root question | Tuscany provinces produce Chianti |
| Number of Words into Root Question | 4 |
| Nominal expansion | provinces, region, territory, area, sub-area |
| Verbal expansion | produce, produced, producing |
| Full Keywords | Tuscany provinces, territory, region, area, sub-area, produce, produced, producing, Chianti |

## 4.2 Passage Retrieval Module

The Passage Retrieval module is responsible for searching web pages (using Google API[1]) and save their full textual content into local files for post-processing. This version of the system is working with 10 downloaded files. In the further versions we intend to work with more files. This module is also responsible for cleaning the HTML files and saving into local files only the content information. After the content is saved into a file, the system will select the relevant sentences based on matching and counting the keywords in the sentences. The number of files that will store relevant information is determined by the strategy being used. If the strategy level is 1st, there is only one file to store the relevant information; if the strategy level is 2nd, there are two files; and if the strategy level is 3th, there are three files. Later in the processing, the quantity of files will affect the number of lists that will be used to build the final list of answers.

The sentences are classified in three classes according to their relevance with respect to the root question. Depending on their classification, the sentences are stored in distinct sets. To demonstrate this stage of processing, we use the example in Table 3, whose number of Keywords is 4 and the Relevance Score is 2 (the half of the number of keywords). In this example, the sentences will be stored in three distinct files according to their level of relevance. t will have "weak" relevance if contains less keywords than the Relevance Score; "medium" relevance is determined if contain identical number of Relevance Score and "strong" if contains more keywords than the Relevance Score. Table 4 shows an example of sentence classification based on number of Keywords.

**Table 4: Sentence classification**

| Sentence | Class |
|---|---|
| During the 1970s producers started to reduce the quantity of white grapes in **Chianti**. | weak |
| Wines labelled **Chianti** Classico come from the biggest **sub-area** of Chianti, that includes the original Chianti heartland. | medium |
| **Chianti** is produced in central **Tuscany region** divided in five **provinces**: Siena, Firenze, Prato, Arezzo and Pistoia. | strong |

## 4.3 Answer Extraction Module

This module performs two main tasks: Candidate Answer Identification and Building the List Answer. Candidate Answer Identification task extracts all words tagged with the proper name tag (in this version of the system we are assuming that all answers are proper names).

The process of Building the List Answer is based on frequency and rules. For the frequency approach, the main elements that will compose the list answer are taken from sentences previously classified as "strong" relevance (which we term Premium List) and will serve to guide the rest of the processing. If we were to consider only these elements, the list of answers would probably contain correct items.

However, the list may be incomplete and lack elements. Then, continuing in the same vein of our strategy, we use the sentences classified as "medium" and "weak" (termed Work List) to confirm and expand the elements in the list. The Final List answers will be composed by elements from Premium List and Work List filtered by thresholds. In addition to the filters, the system uses three Word Occurence Rules based on verb analysis, page title and sentence match. The whole process of building the final list of answers is detailed below:

[1]http://www.google.com.br/cse/

1. The Premium List is built from candidates extracted from the sentences previously classified as "strong" relevance.

2. The Work List is built from candidates extracted from the sentences previously classified with "medium" or "weak" relevance.

3. The elements in the Work List that appear repeated are grouped together and their frequency is calculated.

4. Two frequency thresholds are calculated from the Work List. One threshold will be used to filter the Premium List and the other to filter the Work List. The thresholds are calculated by the following procedure:
   Let $wa = \frac{\sum_j c_j \times j}{\sum_j c_j}$ be the weighted average of the elements in the Work List, where $j$ is the frequency and $c_j$ is the frequency of elements with frequency $j$.
   Let $u$ be an (empirically determined) upper bound on the admissible values for $j$, in order to limit the impact of the elements with very high frequencies.
   Let $\hat{c}_{u,j} = min(j, u)$ be the frequency of frequencies bounded by $u$.
   Let $\hat{wa} = \frac{\sum_j \hat{c}_{u,j} \times j}{\sum_j \hat{c}_{u,j}}$ be the weighted average of the elements in the Work List, taking into account frequency of frequencies bounded by $u$.
   To calculate the threshold $t_P = \hat{wa}$ for the Premium List, $u$ is set to 5 after experimentation.
   To calculate the threshold $t_W = \hat{wa}$ for the Work List, $u$ is set to 1 after experimentation.

5. Filter with $t_P$: Candidates in the Premium List are filtered. The frequency of each candidate is compared to the first threshold previously calculated. The candidate pass to the Final List of Answers if the frequency is equal to or greater than the threshold previously calculated.

6. Filter with $t_W$: Candidates in the Work List with a frequency above the second threshold previously calculated are also included in the Final List of Answers.

7. Filter Verb-Rule: Candidates in the Premium List who were not promoted to the Final List of Answers, after Filter $t_P$ was applied, still have a second chance to be included following the criterion of analysis of the Verb: If the sentence in which the candidate occured contains the same verb of the question, then the candidate is included in the Final List of Answers.

8. Filter Title-Rule: All candidates extracted from texts in which the text title matches (i.e. all keywords are present) the root question pass to Final List Answers.

9. Filter Sentence Match-Rule: All candidates extracted from sentences that match the root question pass to Final List Answers.

Figure 2 summarizes the building of the list of answers.

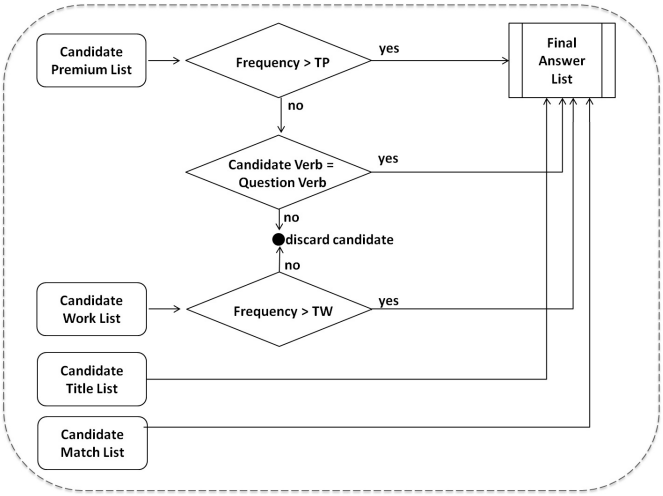


**Figure 2: Building the list of answers.**

## 4.4 Supporting Tools

To build LX-ListQuestion we used the following supporting tools and resources:

- LX-Conjugator[2]: is a tool for conjugation of Portuguese verbs [3]. The system takes an infinitive verb form and delivers the corresponding conjugated forms. The Portuguese verbal inflection is a most complex part of the Portuguese morphology given the high number of conjugated forms for each verb (ca. 70 forms in non pronominal conjugation).

- LX-Suite[3]: is a system for shallow processing of Portuguese [1]. The system is based on a pipeline of several tools. The tools for lemmatization and morphological analysis are inserted at the end of the pipeline and are fed by three other tools: a sentence splitter, a tokenizer and POS tagger.

- LX-Ner[4]: is a tool for recognition of expressions for named entities [5] in Portuguese. The name entities are classified in Persons (PER), Organization (ORG), Location (LOC), Events (EVT) and works (WRK).

- Multi-WordNet PT (MWNPT)[5]: is a lexical semantic network for Portuguese. The database was shaped under the ontological model of wordnets. It spans over 17,200 concepsts/synsets, linked under the semantic relations of hyponymy and hipernymy. These concepts are made of over 21,000 word senses/word forms and 16,000 lemmas. It includes the subontologies under the concepts of Person, Organization, Event, Location and Works of art.

- TEP[6]: Electronic Thesaurus for Brazilian Portuguese [9]. The TEP database stores sets of synonym and antonym for the word forms. We use the database to improve keywords expansion.

[2]http://www.lxcenter.di.fc.ul.pt/services/en/LXServicesConjugator.h
[3]http://www.lxcenter.di.fc.ul.pt/services/en/LXServicesSuite.html
[4]http://lxcenter.di.fc.ul.pt/services/en/LXServicesNer.html
[5]http://lxcenter.di.fc.ul.pt/services/en/LXServicesWordnet.html
[6]http://www.nilc.icmc.usp.br/tep2/

- HTML Parser[7]: HTML Parser is a Java library used to parse HTML. The library allows to transform HTML pages into plain text.

- Google Custom Search[8]: The Google Custom Search is an application programming interface (API) that allows retrieving and displaying search results from Google Custom Search. The API works integrated into the system application. The API provides 100 search queries per day for free.

- Pagico Corpus[9]. The Pagico Corpus is composed by 150 topics in Portuguese where the answers are spread over multiple documents and the topics require multiple answers.

## 4.5 Results

For the experiments we used a set of 10 questions[10] that require List Answers:

Q1: (PT)Instrumentos musicais de origem africana comuns no Brasil. (EN)African musical instruments common in Brazil.
Q2: (PT)Parques do Rio de Janeiro que têm cachoeiras. (EN)Parks of Rio de Janeiro that have waterfalls.
Q3: (PT)Igrejas em Macau. (EN)Churches in Macau.
Q4: (PT)Cidades que fizeram parte do domínio português na Índia. (EN)Cities in India that were under Portuguese rule.
Q5: (PT)Parques nacionais de Moçambique. (EN)National parks in Mozambique
Q6: (PT)Ilhas de Moçambique.(EN)Islands of Mozambique
Q7: (PT)Movimentos culturais surgidos no nordeste do Brasil. (EN)Cultural movements that emerged n the northeast of Brazil.
Q8: (PT)Dioceses católicas de Moçambique. (EN)Catholic dioceses in Mozambique.
Q9: (PT)Candidatos a alguma das eleições presidenciais na Guiné-Bissau. (EN)Candidates for any of the presidential elections in Guinea-Bissau.
Q10: (PT)Capitais das províncias de Angola. (EN)Capitals of the provinces in Angola.

**Table 5: Building Answer List - Results**

| | Premium List | Work List | Answer List | Reference List | Correct |
|---|---|---|---|---|---|
| Q1 | 16 | 9 | 26 | 11 | 4 |
| Q2 | 167 | 89 | 12 | 2 | 1 |
| Q3 | 198 | - | 48 | 17 | 6 |
| Q4 | 100 | 159 | 36 | 22 | 5 |
| Q5 | 236 | 99 | 29 | 4 | 3 |
| Q6 | 89 | - | 39 | 12 | 5 |
| Q7 | 12 | 9 | 21 | 5 | 1 |
| Q8 | 57 | 162 | 34 | 7 | 5 |
| Q9 | 16 | 26 | 19 | 4 | 1 |
| Q10 | 39 | 254 | 21 | 19 | 8 |
| Total | 930 | 807 | 285 | 103 | 39 |

[7]http://htmlparser.sourceforge.net/
[8]http://www.google.com.br/cse/
[9]www.linguateca.pt/Pagico
[10]These questions were based on Pagico: www.linguateca.pt/Pagico

Table 5 shows the results during the building of the answer list. We observe that the number of candidates varies depending on the question. For instance, questions Q1, Q7 and Q9 have far fewer candidates when compared with Q3 or Q5. The candidates of Q3 and Q6 do not have elements in Work List because the level of the strategy was set to 1st.

Among the 103 expected answers in the reference list, the system correctly answered 39 of them. It is important to mention that the LX-ListQuestion System is a dynamic system running over the Web and there is no guarantee that all answers can be found there.

Table 6 shows the evaluation of the LX-ListQuestion System. The metrics used are: recall, precision and F-measure. These metrics take into consideration two lists: a reference list of correct answers and the system list (answers returned by the QA system).

Precision: C is the number of common elements between reference and system lists and S is the number of elements given by the system.

$Precision = \frac{C}{S}$

Recall: C is the number of common elements between reference and system lists and L is the number of elements in reference list.

$Recall = \frac{C}{L}$

F-measure: its the combination (harmonic mean) between Recall and Precision.

$F - measure = \frac{2*Recall*Precision}{Recall+Precision}$

**Table 6: Metric Evaluation**

| Question | Precision | Recall | F-Measure |
|---|---|---|---|
| Q1 | 0.15 | 0.36 | 0.21 |
| Q2 | 0.08 | 0.50 | 0.14 |
| Q3 | 0.13 | 0.35 | 0.18 |
| Q4 | 0.14 | 0.23 | 0.17 |
| Q5 | 0.10 | 0.75 | 0.18 |
| Q6 | 0.13 | 0.42 | 0.20 |
| Q7 | 0.05 | 0.20 | 0.08 |
| Q8 | 0.15 | 0.71 | 0.24 |
| Q9 | 0.05 | 0.25 | 0.09 |
| Q10 | 0.38 | 0.42 | 0.40 |
| AVERAGE | 0.14 | 0.38 | 0.20 |

We observe from Table 6 that the system answered all questions. It achieved better recall for the questions Q5 and Q8. The Question Q10 obtained better precision and also f-measure. Overall, the system scores 0.38 of recall, which is a very competitive result for the current state-of-art. Exploring the redundancy of information seems to be a good approach to this task, but it alone cannot handle all problems.

## 4.6 User interface

LX-ListQuestion is available on the web:
*http://nlxserv.di.fc.ul.pt/lxlistquestion/index.jsp*
In our tests, the response time ranged between 16 and 28 seconds of processing from submitting the question and getting the list of answers. We chose to use word cloud as the form of presentation of the results because, together with the final answers list, it provides a visual representation of related words. The word cloud helps the user to understand the context in which the answers may be embedded. Figure 3 shows LX-ListQuestion System online GUI.

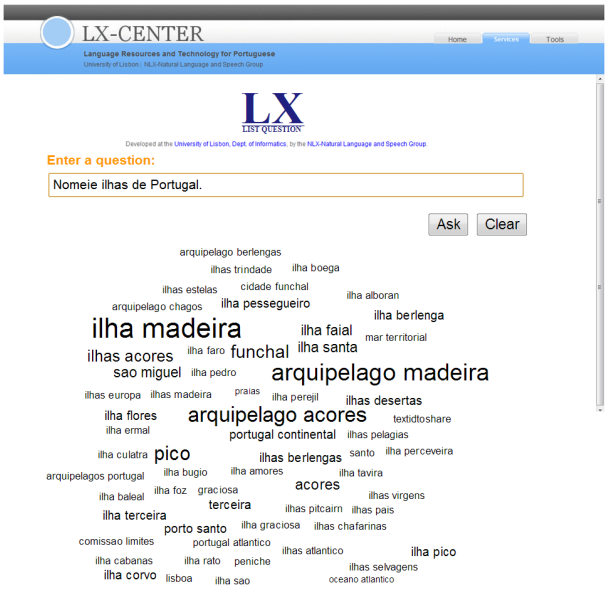


**Figure 3: LX-ListQuestion online GUI.**

## 5. CONCLUDING REMARKS

LX-ListQuestion is a fully-fledge Web based QA system that generates answers to list questions and presents them in a word cloud. The system exploits the redundancy of information available in the Web and combines with word occurrence rules to improve QA accuracy. This version handles Portuguese. The next version will be extended to provide answers to other languages as well. Our system gets a competitive results.This work is being developed as the subject of a progressing doctoral thesis and new improvements will be implemented.

## 6. REFERENCES


[1] A. Branco and J. R. Silva. A suite of shallow processing tools for portuguese: Lx-suite. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, EACL '06, pages 179–182, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[2] N. Cardoso, D. Batista, F. J. López-Pellicer, and M. J. Silva. Where in the wikipedia is that answer? the xldb at the gikiclef 2009 task. In C. Peters, G. M. D. Nunzio, M. Kurimo, D. Mostefa, A. Peñas, and G. Roda, editors, *CLEF*, volume 6241 of *Lecture Notes in Computer Science*, pages 305–309. Springer, 2009.

[3] F. Costa. Verbal conjugation in portuguese. Internal report, University of Lisbon, Department of Informatics, 2004.

[4] I. Dornescu. Semantic qa for encyclopaedic questions: Equal in gikiclef. In C. Peters, G. M. D. Nunzio, M. Kurimo, D. Mostefa, A. Peñas, and G. Roda, editors, *CLEF*, volume 6241 of *Lecture Notes in Computer Science*, pages 326–333. Springer, 2009.

[5] E. Ferreira, J. Balsa, and A. Branco. Combining rule-based and statistical models for named entity recognition of Portuguese. In *In Proceedings of Workshop em Tecnologia da Informação e de Linguagem Natural*, pages 1615–1624, 2007.

[6] R. J. Gaizauskas, M. A. Greenwood, H. Harkema, M. Hepple, H. Saggion, and A. Sanka. The university of sheffield trec 2005 q&a experiments. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, Gaithersburg, Maryland, November 15-18, 2005*, volume Special Publication 500-266. National Institute of Standards and Technology (NIST), 2005.

[7] S. Hartrumpf and J. Leveling. GIRSA-WP at GikiCLEF: Integration of structured information and decomposition of questions. In *10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30-October 2, Revised Selected Papers*, Lecture Notes in Computer Science (LNCS). Springer, 2010. (to appear).

[8] A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. Question answering with lcc's chaucer at trec 2006. In *TREC*, 2006.

[9] E. G. Maziero, T. A. S. Pardo, A. Di Felippo, and B. C. Dias-da Silva. A base de dados lexical e a interface web do tep 2.0: Thesaurus eletrônico para o português do brasil. In *Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, WebMedia '08, pages 390–392, New York, NY, USA, 2008. ACM.

[10] M. Razmara and L. Kosseim. Answering list questions using co-occurrence and clustering. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*. European Language Resources Association, 2008.

[11] T. Strzalkowski and S. Harabagiu. *Advances in Open Domain Question Answering*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[12] E. W. D. Whittaker, J. R. Novak, P. Chatain, and S. Furui. Trec 2006 question answering experiments at tokyo institute of technology. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, November 14-17, 2006*, volume Special Publication 500-272. National Institute of Standards and Technology (NIST), 2006.

[13] M. Wu and T. Strzalkowski. Utilizing co-occurrence of answers in question answering. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics, 2006.

[14] H. Yang and T.-S. Chua. Effectiveness of web page classification on finding list answers. In M. Sanderson, K. Järvelin, J. Allan, and P. Bruza, editors, *SIGIR*, pages 522–523. ACM, 2004.

[15] H. Yang, H. Cui, M. Maslennikov, L. Qiu, M.-Y. Kan, and T.-S. Chua. Qualifier in trec-12 qa main task. In *TREC*, pages 480–488, 2003.