Query Classification and Expansion in Just.Ask Question Answering System

Ricardo Pires

IST - Instituto Superior Técnico
rmpp@ist.utl.pt

Abstract This thesis focus is the Just.Ask Question Answering system developed at L2F, having as main goals to endow Just.Ask with Query Classification and Query Reformulation/Query Expansion mechanisms in order to, respectively, allow the users to be able to express their information needs in form of queries and to improve the number of retrieved relevant passages. Regarding the first goal, a Support Vector Machines query classifier was trained and tested using features developed for the Query Classification task that take into consideration the main characteristics of queries, obtaining for coarse and fine-grained classification a maximum accuracy of 86% and 80.2%, respectively. Additionally, two approaches are proposed to perform simultaneous Query and Question Classification, with the best approach having a maximum accuracy of 86.6% and 79.7% for coarse and fine-grained classification, respectively. As for the second goal, two query formulators that implement Query Reformulation and Query Expansion mechanisms, respectively, were devised. The first one produces queries that are possible reformulations of the user question and that are obtained by matching the user question with a set of 163 regular expressions. The second one expands the question headword/compound headword and the main verbs that exist in the user question with semantically related terms obtained using Wordnet. When using Bing and Lucene search engines there is an improvement over the baseline retrieval performance of 17 questions (9.2%) and 20 questions (10.9%) over the number of questions with positive passages, respectively.

1 Introduction

Question Answering (QA) systems allow users to express their information needs using a natural language question and return an exact answer to that question. Just.Ask is a QA system developed at L2F, that combines rule- with machine learning-based components and uses several state of the art strategies in QA.

Question Classification which consists in assigning to the user question a predefined semantic category that represents the expected semantic type of the answer, is one strategy used by QA systems (including Just.Ask) with multiple purposes. These purposes often include: defining some of the query terms; determining the information source where the passages for answering a question are obtained and creating specific answer extraction techniques according to the question type.

Question classification is based on the assumption that users express their information needs using syntactically well formed questions to do so. However, users usually perform their questions in form of queries constituted by a short number of terms. This user behavior must be taken into account by QA systems by allowing users to express their information needs in form of queries. Previous to this work, Just.Ask could not classify queries since it was only endowed with the necessary mechanisms to classify questions. Therefore, it is of the utmost importance the utilization of both Question Classification and Query Classification mechanisms.

To obtain the exact answer to a question, QA systems (including Just.Ask) use Information Retrieval (IR) techniques. IR techniques filter out relevant passages from web pages or documents, to narrow down the search for possible answers. Answers that are not contained
in these passages are not considered by the system, so it is of the utmost importance to retrieve as much relevant passages as possible. Query Reformulation and Query Expansion (which is a specification of query reformulation) are solutions to improve the number of relevant passages retrieved by a QA system.

The goals of the presented work are to create Query Classification and Query Expansion mechanisms for Just.Ask. Question Classification in Just.Ask attains state of the art results and a detailed description of the impact of several features that led to these results is given in [9]. So, the first goal of this work is dedicated to understand how Query Classification can be accomplished in Just.Ask, in order to enable users to express their information needs in form of queries. A research of the literature regarding Query Classification is conducted in this thesis with that purpose. The second goal consists in improving the Query Formulator module of Just.Ask by adding new Query Reformulation and Query Expansion techniques, so that more relevant passages can be retrieved by the system. In order to accomplish this goal, a research of the existing Query Reformulation/Query Expansion techniques as well as the Query Reformulation/Query Expansion techniques that are used by existing QA systems, is performed.

2 State of the Art

This section is divided into two subsections. The first part focuses on the Query Classification state of art, depicting the main challenges in this task and the approaches developed to solve them. The second subsection is devoted to the Query Reformulation/Query Expansion state of art, describing the existing Query Reformulation/Query Expansion techniques and the Query Reformulation/Query Expansion techniques that are currently in use by existing QA systems that were relevant to this work.

2.1 Query Classification

The main obstacle in query classification is to deal with the lack of information inherent in a query due to small number of word that queries contain. Thus, researchers have devoted their efforts in studying possible ways to enhance queries with additional information. To this work’s scope, the most relevant approach is enhancing queries based on initial search results or trough query expansion using knowledge structures.

The 2005 KDD-Cup on Web query classification inspired a whole new approach to query classification, that focus on enriching queries using Web search engines and directories. The KDD-Cup was an IR contest in which the participants were provided with a small taxonomy (67 categories) along with a set of labeled queries, and would have to use this training data to build a query classifier. The winning solution of the KDD-Cup [8] uses an ensemble\(^1\) of classifiers in conjunction with multiple search engines. More specifically, each query to be classified is enriched by its related Web pages together with their category information, collected trough various search engines. After this query enrichment phase, two base classifiers are developed - the synonym-based and statistical-based classifier. Two ensemble classifiers are obtained by combining these two kinds of classifiers according to different ensemble strategies. This improves the overall classification performance significantly more than the base classifiers.

2.2 Query Expansion and Query Reformulation

Query Reformulation and Query Expansion techniques can be used to improve the quality of a query in a QA system. Query reformulation is defined as “an attempt to improve poor users queries by: removing terms that deteriorate the retrieval performance, adding terms

\(^1\) An ensemble is a set of models whose predictions are combined by weighted averaging or voting.
that aid retrieval, reweighting existing or new query terms to give the query a different emphasis, or a combination of those methods\[1\]. Query expansion (which is a specification of query reformulation) is a simpler approach where “the initial query is refined by adding new terms, rather than removing or replacing original terms, or altering their weights” \[1\]. Query expansion can be performed in two different ways - iterative and automatic expansion. Iterative expansion methods are characterized by improving the query with terms selected by the system user. In automatic expansion methods the terms utilized for expansion are selected automatically (without the user intervention) by the system using several strategies. These strategies include: usage of knowledge structures containing terms for expansion such as Wordnet \[2\], the best N-ranked terms of the initial search result \[10\], terms from past queries performed to the system that are associated with the documents in the collection \[1\], iteratively selecting and combining basic feedback methods \[4\], etc.

From the query expansion/reformulation techniques in the analyzed QA systems, Open Ephyra \[6,7\] and Raposa \[5\] were the most relevant to this work. Regarding query reformulation, Open Ephyra uses queries that are possible reformulations of the user question and that try to anticipate the location of the answer in order to extract it. As for query expansion, Open Ephyra uses queries in which question terms are expanded with semantically related using knowledge structures to do so (more specifically Wordnet). Similarly to Open-Ephyra, Raposa expand the verbs existing in his queries using a knowledge structure, more specifically a verb thesaurus solely developed for that purpose.

\section{Query Classification}

This section is devoted to the creation of Query Classification mechanisms in Just.Ask system and it is divided into three subsections. The first part, Subsection 3.1 describes the created methods for converting questions in the training and test datasets into queries. Subsection 3.2 describes the features developed for the Query Classification task and the results obtained by the devised Query Classifier classifier. Finally, Subsection 3.3 depicts the created approaches to perform simultaneous Question and Query Classification, as well as the results obtained in the evaluation of this task.

\subsection{Adapting the training and test datasets}

Just.Ask models Question Classification using a SVM model trained and tested with a dataset of 5500 and 500 questions respectively, classified according to the Li and Roth taxonomy. In order to use the same datasets for Query Classification, the questions in those datasets need to be transformed into queries that represent accurately the queries that a user would do for those questions. This task will be performed using two methods, from now on - the Just and Regex methods.

The Just method is currently implemented in Just.Ask and consists in substituting the questions of the training and tests sets by the queries that Just.Ask generates for those questions. The training and tests questions sets are used as input for the system and the Keyword Query Formulator of Just.Ask, as the name suggests, uses a keyword oriented approach to transform each question into a query by removing question words, stop-words, function words and useless punctuation \textsuperscript{2}. After the query has been created, Just.Ask substitutes the respective question in the respective question set, for the query that was created.

The observation of several queries produced by this method showed us that some of those queries would not suitable examples of queries submitted by users to Just.Ask. The Regex method addresses this issue by attempting to take into consideration how users formulate their queries. The Regex method performs a matching between questions and a set of 100 regular expressions, that transform those questions into queries by performing one, or both

\textsuperscript{2} Text between quotes remains unchanged due to its importance for the query target.
of the following actions: question words and the most common words that follow them are removed and some set of words are substituted by one word that a user would probably use instead of that set of words.

3.2 Feature Engineering in Query Classification

The Query Classification task using the datasets obtained with the Regex and Just methods, is performed using the LIBSVM implementation of a Support Vector Machines (SVM) model with linear kernel and using the one-versus-all-multi-class-approach. As features for the Query Classification task, some of the features used for Question Classification are utilized - n-grams, binary n-grams and POS tags - as well as some features that were developed for this specific task and that take into account the main characteristics of queries:

- Number of Tokens - Measures the query length with its number of tokens;
- Number of Chars - Measures the query length with the number of characters that compose the query’s tokens;
- Binary Length - Indicates if the query is short or long. A query is considered short if it has less than five tokens;
- Word Shape - Indicates the number of query’s tokens that are lowercased, uppercased, have the first character capitalized, are only constituted by digits, or are different from any of the previous cases;
- Binary Word Shape - Indicates if a certain query’s tokens belongs to any of the previous “shape” categories;
- Pseudo Headword - It is considered that the first noun in the query represents the query’s headword. This consideration may not be valid for every query, which led to this feature designation;
- First Token - A considerable number of queries have two up to four tokens. In those cases, it may be expectable that the query’s first token represents the information that the user soughs for.

3.2.1 Coarse-Grained Classification Results

Table 1 presents, for coarse-grained classification, the most relevant results that ultimately lead to the best feature combination for both Just and Regex methods.

When considering each feature in isolation, Unigrams, Binary Unigrams and Ner-Replace features clearly stand out as the most discriminating features. More specifically, Binary Unigrams and Ner-replace are the best individual features for Just and Regex methods, yielding an accuracy of 74% and 84.2%, respectively.

When considering the combination of two different features only, both the combination of Binary Word Shape with Ner-replace or Unigrams with First Token, achieve the best results for the Just method with an accuracy of 75.2%. On the other hand, the combination of Ner-replace with First Token attains the best results for the Regex method with a accuracy of 85.6%. One intriguing fact is that, although Binary Unigrams showed a slightly better performance over Unigrams when both are used as individual features, the combination of Unigrams with other features produces equal or slightly better results over the combination of Binary Unigrams with those same features. This fact indicates that the use of Unigrams may be more beneficial for query classification than the use of Binary Unigrams.

The combination of more than two different features leads, not surprisingly, to the best accuracy results for both Just and Regex method. The combination of Unigrams, Binary Word Shape and First Token attains an accuracy of 78.6% for the Just method, while the combination of Unigrams, POS tags and First Token yields an accuracy of 86% for the Regex method. The fact that Unigrams are part of the best feature combinations corroborates the hypothesis elaborated in the previous paragraph.

From this analysis we can conclude from the set of features developed specifically for query classification, the Binary Word Shape and First Token features are the most useful features for coarse-grained classification.
### Accuracy for coarse-grained classification

<table>
<thead>
<tr>
<th>Features</th>
<th>Regex Method</th>
<th>Just Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Token</td>
<td>48%</td>
<td>64%</td>
</tr>
<tr>
<td>Binary Word Shape</td>
<td>35.6%</td>
<td>36.4%</td>
</tr>
<tr>
<td>POS tags</td>
<td>39.4%</td>
<td>38.2%</td>
</tr>
<tr>
<td>Ner-Replace</td>
<td>83.8%</td>
<td>74%</td>
</tr>
<tr>
<td>Unigrams</td>
<td>83.8%</td>
<td>71.8%</td>
</tr>
<tr>
<td>Binary Unigrams</td>
<td>84.2%</td>
<td>72%</td>
</tr>
<tr>
<td>Bigrams</td>
<td>59.2%</td>
<td>33%</td>
</tr>
<tr>
<td>First Token + Ner-Replace</td>
<td>85.6%</td>
<td>74.8%</td>
</tr>
<tr>
<td>Binary Word Shape + Ner-Replace</td>
<td>83.8%</td>
<td>75.2%</td>
</tr>
<tr>
<td>Unigrams + Binary Word Shape</td>
<td>83%</td>
<td>75.2%</td>
</tr>
<tr>
<td>Unigrams + First Token</td>
<td>85.2%</td>
<td>74.6%</td>
</tr>
<tr>
<td>Unigrams + POS tags</td>
<td>85%</td>
<td>72.8%</td>
</tr>
<tr>
<td>Binary Unigrams + Binary Word Shape</td>
<td>82.8%</td>
<td>74.6%</td>
</tr>
<tr>
<td>Binary Unigrams + First Token</td>
<td>85%</td>
<td>75%</td>
</tr>
<tr>
<td>Binary Unigrams + POS tags</td>
<td>85%</td>
<td>72.4%</td>
</tr>
<tr>
<td><strong>Unigrams + Binary Word Shape + First Token</strong></td>
<td><strong>84.6%</strong></td>
<td><strong>78.6%</strong></td>
</tr>
<tr>
<td><strong>Unigrams + POS tags + First Token</strong></td>
<td><strong>86%</strong></td>
<td><strong>74.6%</strong></td>
</tr>
</tbody>
</table>

*Table 1: Accuracy of Just and Regex methods for coarse-grained classification.*

### 3.2.2 Fine-Grained Classification Results

Similarly to the previous subsection, Table 2 presents for fine-grained classification the most relevant results that ultimately lead to the best feature combination for both Just and Regex methods.

<table>
<thead>
<tr>
<th>Features</th>
<th>Regex Method</th>
<th>Just Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Token</td>
<td>34.6%</td>
<td>36.4%</td>
</tr>
<tr>
<td>Ner-Replace</td>
<td>78.8%</td>
<td>68.6%</td>
</tr>
<tr>
<td>Binary Word Shape</td>
<td>33.2%</td>
<td>33.6%</td>
</tr>
<tr>
<td>Number of Tokens</td>
<td>32%</td>
<td>30.8%</td>
</tr>
<tr>
<td>Unigrams</td>
<td>78.4%</td>
<td>67.6%</td>
</tr>
<tr>
<td>Binary Unigrams</td>
<td>78.4%</td>
<td>67.6%</td>
</tr>
<tr>
<td>Bigrams</td>
<td>50%</td>
<td>23%</td>
</tr>
<tr>
<td><strong>Binary Word Shape + Ner-Replace</strong></td>
<td><strong>80.2%</strong></td>
<td><strong>69.6%</strong></td>
</tr>
<tr>
<td>Unigrams + Ner-Replace</td>
<td>79%</td>
<td>66.4%</td>
</tr>
<tr>
<td>Unigrams + Binary Word Shape</td>
<td>78.6%</td>
<td>69.2%</td>
</tr>
<tr>
<td>Unigrams + Number of Tokens</td>
<td>78.2%</td>
<td>68.2%</td>
</tr>
</tbody>
</table>

*Table 2: Accuracy of Just and Regex methods for fine-grained classification.*

As in coarse-grained classification, when each feature is considered in isolation, Unigrams, Binary Unigrams and Ner-Replace features stand out as the most discriminating features. Ner-Replace is the best individual feature for the Just method with an accuracy of 68.6%, while Unigrams and Binary Unigrams are the best individual features for the Regex method yielding an accuracy of 78.4%.

As opposed to coarse-grained classification, the combination of only two different features leads to the best accuracy results for both Just and Regex method. More specifically, the combination of Binary Word Shape with Ner-Replace is the best combination for Just and Regex methods, yielding an accuracy of 69.6% and 80.2%, respectively. No results of the
combination of Binary Unigrams with other features are shown, since these results are always equal or inferior to the results attained by the combination of Unigrams with other features. This confirms that the use of Unigrams is indeed more beneficial for query classification than the use of Binary Unigrams.

From this analysis we can conclude that from the set of features developed specifically for query classification, the Binary Word shape is the most useful feature for fine-grained classification.

3.3 Simultaneous Query and Question Classification

All the work described in the previous section was devised with the purpose of enabling Just.Ask to receive user queries as input. At this moment, we have classifiers solely dedicated to the classification of questions and queries, respectively. Due to the fact that users will be able to use two types of input (queries and questions), the previous mentioned classifiers cannot be used, in a direct fashion, since the input type that is going to be submitted by the users is not known beforehand.

3.3.1 Approaches to Simultaneous Query and Question Classification

To solve the former problem two approaches are considered: devising a classifier that can classify both questions and queries, and finding distinct characteristics between queries and questions, so that Just.Ask can use according to the input type, the most adequate classifier (query or question classifier).

Regarding the first approach, from which results the classifier denoted as “Query and Question Classifier”, a SVM classifier with a linear kernel is trained with a dataset of 5500 queries and 5500 questions. The queries in the train dataset correspond to the questions of the train dataset used to train Just.Ask’s question classifier converted into queries by the Just method. The questions in the train dataset used in the evaluation of Just.Ask’s question classifier, are also the same questions used in the train dataset of the Query and Question Classifier.

The second approach takes advantage of the previous developed classifiers for queries and questions, as it selects the most adequate classifier for the type of input submitted by the user. Due to this characteristic we denote the classifier produced by this approach as the “Most Adequate Query and Question Classifier”. The former process is accomplished by taking advantage of two characteristics to distinguish the possible two types of input: the presence of a question word in the beginning of the input and the length (in number of terms) of the input itself. These two characteristics are used in a simple procedure: if the user input starts with a question word and it has more than K terms, the input is considered to be a question and we classify that input with the best question classifier. Otherwise, the user input is considered to be a query and we classify that input with the best query classifier.

3.3.2 Coarse and Fine-Grained Classification Results

The presented approaches were evaluated using a test dataset of 500 queries and 500 questions. The questions and queries in this dataset correspond, respectively, to the questions of the test dataset used in the evaluation of Just.Ask’s Question Classifier and to these same questions converted into queries by the Just method. Additionally, both the query classifier devised in the present section and Just.Ask’s Question Classifier were evaluated using the same test dataset, in order to have an assessment of the performance of each classifier when trying to classify queries and questions simultaneously.

Table 3 presents for coarse and fine-grained classification the accuracy of all considered classifiers.

\(^\text{3}\) We consider K to be the number of terms of the lengthiest query in the training dataset used in the first approach.
### Accuracy for fine and coarse-grained classification

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>Coarse-Grained</th>
<th>Fine-Grained</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Classifier</td>
<td>81.9%</td>
<td>74.7%</td>
</tr>
<tr>
<td>Question Classifier</td>
<td>80.3%</td>
<td>73.4%</td>
</tr>
<tr>
<td>Query and Question Classifier</td>
<td>82.7%</td>
<td>76.3%</td>
</tr>
<tr>
<td>Most Adequate Query and Question Classifier</td>
<td>86.6%</td>
<td>79.7%</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of simultaneous query and question coarse-grained classification.

As it can be seen, the Most Adequate Query and Question Classifier obtains the best results with an accuracy of 86.6% and 79.7% for coarse and fine-grained classification, respectively. This fact indicates that distinguishing the user input into queries or questions and redirecting this input to its most appropriate classifier, is indeed a good approach for classifying queries and questions simultaneously.

From the remaining classifiers the Query and Question Classifier attains the best performance for coarse and fine-grained classification yielding, respectively, 82.7% and 76.3% of accuracy when using Ner-Replace, Semantic Headword and First Token as features. This outcome was expected since the latter classifier was trained with both queries and questions, as opposed to the Best Query and Question classifiers which were trained only with queries and questions, respectively, and therefore should not have the same performance as a classifier trained with both queries and questions. Although this last statement is valid, these three classifiers have all similar performances for both coarse and fine grained classification, with a variation of less than 2.7% between their accuracies.

From this analysis we can conclude that the Most Adequate Query and Question Classifier is the best classifier for classifying queries and questions simultaneously. We can also conclude that the Best Query and Best Question Classifiers, which are SVM classifiers with linear kernels trained with queries and questions respectively, have almost the same performance as a SVM with linear kernel trained with both queries and questions.

## 4 Query Reformulation and Query Expansion

This chapter is devoted to the creation of Query Reformulation/Query Expansion mechanisms in Just.Ask system and it is divided into four subsections. Subsections 4.1 and 4.2 describe two query formulators - the Query Reformulation and Query Expansion Formulators - that implement Query Reformulation and Query Expansion mechanisms, respectively. Subsection 4.3 depicts the experimental setup for the evaluation results described in Subsections 4.4 e 4.5 for the individual and combined use of the devised query formulations when using Bing and Lucene search engines, respectively.

### 4.1 Query Reformulation Formulator

The Query Reformulation Formulator was inspired by Open-Ephyra and takes advantage of the large data redundancy present in information sources (such as the Web), where an answer to a certain question is stated in multiple documents and in multiple formulations, by producing queries that are possible reformulations of the user question. The reformulations are obtained matching the user question with a set of regular expressions as follows: first, the question is matched with a set of 150 regular expressions that are specific to the category that was assigned to that question by Just.Ask’s Question Classifier with the intent of taking advantage of some vocabulary that is specific to that certain category, by using that vocabulary to generate more queries with reformulations of the question. If no match occurs, the question is matched with a set of 13 regular expressions that take into account the most common formulations that questions have. If no match still occurs, the Keyword and Regex Query Formulators are used to transform the user question into a query.
4.2 Query Expansion Formulator

The Query Expansion Formulator was inspired by Open-Ephyra and Raposa, and produces queries in the Disjunctive Normal Format where the question headword/compound headword and the main verbs that exist in the user question are expanded with semantically related terms, obtained using Wordnet. Just.Ask uses the Java WordNet Library (JWNL) extension, JWNLSimple [3], developed by L2F to leverage the Wordnet resources, using the following procedures:

- Direct Hypernym - Retrieves direct hypernyms of the given query term. For example, “red” is a direct hypernym of “vermilion”;
- Similar To - Retrieves words that possess a similar meaning to the given query term. For instance, “serene” is similar to “calm”.

4.3 Experimental Setup

To evaluate the performance each formulator and the performance obtained when using both formulators simultaneously, Just.Ask was ran using a corpus with 183 factoid questions and two different corpora for answers: the Web and paragraphs from the New York Times from 1987 to 2007, indexed by Lucene. For these two corpora, Bing and Lucene are respectively used as search engines to retrieve relevant passages from them. Bing search API allows a developer to retrieve at most 50 passages per request and by manipulating the offset parameter up to 1000 passages per query can be obtained. On the other hand, a conducted preliminary experiment with Lucene using the testing data showed that Lucene tends to retrieve between 40/60 passages per query. Thus, in order to evaluate both search engines under the same conditions, we chose to use 50 passages as the maximum number of passages to retrieve per query. Finally, the following evaluation metrics were used:

- \(\#\text{Question}_{1+}\text{PosPassage}\) - Number of passages that contain at least one correct answer;
- \(\text{MRR}_{\text{ALLQ}}\) - Mean reciprocal rank of the first positive passage for all questions;
- \(\text{MRR}_{Q\text{POSPASSAGES}}\) - Mean reciprocal rank of the first positive passage only for the questions for which at least one positive passage exists.

4.4 Query Reformulation and Expansion Combination Bing Results

Table 4 presents, for Bing search engine, the best results obtained individually by using each one of the query formulators, the Just.Ask’s baseline retrieval results for comparative purposes and more importantly, the results obtained by the combination of the best parameterizations of each query formulator. The combinations that were considered are:

- Bing Combination 1 - Query Reformulation Formulator with Keyword Query Formulator + Query Expansion Formulator (Main Verbs expansion) with Keyword Query Formulator using all the retrieved expansion terms.
- Bing Combination 2 - Query Reformulation Formulator with Regex Query Formulator + Query Expansion Formulator (Main Verbs expansion) with Keyword Query Formulator using all the retrieved expansion terms.

It can be seen that results obtained by the combination of the best parameterizations of both Query Reformulation and Query Expansion Formulators improve, for all the evaluation metrics, the results obtained when using each one of those formulators individually. The results of both combinations are identical except in the \(\#\text{Question}_{1+}\text{PosPassage}\) where one more question with at least one positive passage is retrieved by Bing Combination 1 over the \(\#\text{Question}_{1+}\text{PosPassage}\) attained by Bing Combination 2. Therefore, Bing Combination 1 stands out as being Just.Ask’s best Query Reformulation and Expansion combination, and more importantly, leads to Just.Ask’s best retrieval performance when using Bing as search engine. These results represents an improvement, over the baseline retrieval performance, of 17 questions (9.2%) in the \(\#\text{Question}_{1+}\text{PosPassage}\), 0.15 in the \(\text{MRR}_{\text{ALLQ}}\) and 0.11 in the \(\text{MRR}_{Q\text{POSPASSAGES}}\).
Table 4: Query Reformulation and Expansion combination results when using Bing search engine.

Table 5: Query Reformulation and Expansion combination results when using Lucene search engine.

The first consideration than comes to mind from analyzing the table is that results obtained by the combination of the best parameterizations of both Query Reformulation

---

4.5 Query Reformulation and Expansion Combination Lucene Results

Table 5 presents, for Lucene search engine, the best results obtained individually by using each one of the query formulators, Just.Ask’s baseline retrieval results for comparative purposes and more importantly the results obtained by the combination of the best parameterizations of each query formulator. The combinations that were considered are:

- Lucene Combination 1 - Query Reformulation Formulator with Keyword Query Formulator + Query Expansion Formulator (HeadVerb expansion) with Keyword Query Formulator using all the retrieved expansion terms.
- Lucene Combination 2 - Query Reformulation Formulator with Regex Query Formulator + Query Expansion Formulator (HeadVerb expansion) with Keyword Query Formulator using all the retrieved expansion terms.

---

Footnote: HeadVerb stands for the simultaneous expansion of the question headword/compound headword and the main verbs in a query.

---
and Query Expansion Formulators are identical for all evaluation metrics, and they only surpass the best results attained by the other formulators in the #Question1+PosPassage evaluation metric. Specifically, there is an improvement of seven questions (3.8%) over the best value of #Question1+PosPassage obtained when using the Query Expansion Formulator with the Keyword Query Formulator. Since both combinations of the best parameterizations also outperform the Query Expansion Formulator with the Keyword Query Formulator in the other evaluation metrics, we can state that both combinations outperform the latter formulator.

When considering the best results of the Query Reformulation Formulator for the MRR_{ALLQ} and MRR_{QPOSPASSAGES} evaluation metrics, obtained when using the Keyword Query Formulator, both combinations suffer a decrease of 0.01 and 0.07 respectively. We consider this observed decrease in the MRR_{ALLQ} and MRR_{QPOSPASSAGES} values of both combinations of the best parameterizations, to be of less relevance when compared to the improvement of 14 questions (7.6%) in the #Question1+PosPassage since it results, in the best case scenario, in 14 more correct questions answered. Therefore, we consider that Lucene Combination 1 and Lucene Combination 2 outperform both Query Reformulation and Query Expansion Formulators leading to Just.Ask’s best retrieval performance when using Lucene as search engine. The results of both combinations represent an improvement, over the baseline retrieval performance of 20 questions (10.9%) in the #Question1+PosPassage and 0.03 in the MRR_{ALLQ}.

5 Conclusion

Just.Ask is a QA system developed at L2F, that combines rule- with machine learning-based components and uses several state of the art strategies in QA. The main goals of this thesis were to endow Just.Ask with Query Classification and Query Reformulation/Query Expansion mechanisms in order to, respectively, allow the users to be able to express their information needs in form of queries and to improve the number of retrieved relevant passages.

Regarding the first goal, the Regex and Just Methods were created to convert the questions in the training and test datasets used in the Question Classification task, into queries that attempt to accurately represent the queries that a user would do for those questions. A SVM query classifier was trained and tested with the converted datasets, using features developed for the Query Classification task that take into consideration the main characteristics of queries. The SVM query classifier obtains for coarse-grained classification a maximum accuracy of 86% when using the Unigrams, First Token and POS tags features, and obtains for fine-grained classification a maximum accuracy of 80.2% when using the Binary Word Shape and Ner-Replace features.

Finally, since Just.Ask is able to receive queries and questions as user input due to all the mentioned contributions, two approaches were devised to allow Just.Ask to perform simultaneous Query and Question Classification. The best approach differentiate queries from questions based in the presence of a question word in the beginning of the input and the length (number of terms) of the input itself, redirecting the input to the best Query Classifier if the input is considered to be a query, or to the best Question Classifier otherwise. This approach attains a maximum accuracy of 86.6% and 79.7% for coarse and fine-grained classification, respectively, when tested using a dataset with both queries and questions.

As for the second goal, two query formulators that implement Query Reformulation and Query Expansion mechanisms, respectively, were devised - the Query Reformulation and Query Expansion Formulators. The Query Reformulation Formulator takes advantage of the large data redundancy present in information sources (such as the Web), where an answer to a certain question is stated in multiple documents and in multiple formulations, by producing queries that are possible reformulations of the user question. The reformulations are obtained matching the user question with a set of 163 regular expressions that take into
account specific question formulations of fine-grained categories, as well as more common
deductive Normal Format where the question headword/compound headword and the
and the main verbs that exist in the user question are expanded with semantically related terms,
obtained using Wordnet.

Finally, the use of both of the previous query formulator simultaneously leads to the
best retrieval performance attained by Just.Ask. When using Bing search engine there
is an improvement over the baseline retrieval performance, of 17 questions (9.2%) in the
\#Question\_{1+PosPassage}, 0.15 in the MRR\_{ALLQ} and 0.11 in the MRR\_{QPOS\_PASSAGES}. For
Lucene search engine the observed improvements over the baseline retrieval performance are
20 questions (10.9%) in the \#Question\_{1+PosPassage} and 0.03 in the MRR\_{ALLQ}.

6 Future Work

In future work it would be interesting to:

- Study methods to successfully identify the headword of a query so it can be used as
  feature for the Query Classification task;
- Evaluate the proposed method of distinguish queries from questions in order to access
  the need of devising more accurate methods to do so;
- Improve the reformulation regular expressions (answer patterns) of the Query Reformu-
  lation Formulator by automatically learning new answer patterns using question-answer
  pairs as training data;
- Create methods to evaluate the usefulness of the retrieved expansion terms to the query
  at hand so that only the most useful are used to expand the query;
- Perform query expansion using other resources besides Wordnet such as UBY\textsuperscript{5}.

References

3. P. Fialho, S. Curto, A. Mendes, and L. Coneur. Wordnet framework improvements for nlp:
   Defining abstraction and scalability layers. Technical Report 8113, Technical University of
   Lisbon - Spoken Language Systems Lab, September 2011.
4. Yuanhua Lv, ChengXiang Zhai, and Wan Chen. A boosting approach to improving pseudo-
   relevance feedback. In Proceedings of the 34th international ACM SIGIR conference on Research
   and development in Information, pages 165–174, New York, NY, USA, 2011. ACM.
5. Luis Sarmento. Experiments with query expansion in the RAPOSA (FOX) question answering
6. Nico Schlaefer, Petra Gieselmann, Thomas Schaaf, and Alex Waibe. A pattern learning ap-
   proach to question answering within the ephyra framework. In Text, Speech and Dialogue, pages
7. Nico Schlaefer, Jeongwoo Ko, Justin Betteridge, Manas A. Pathak, Eric Nyberg, and Guido
8. Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang
9. J. Silva, L. Coheur, A. Mendes, and A. Wichert. From symbolic to sub-symbolic information
10. Jinx Xu and W. Bruce Croft. Improving the effectiveness of information retrieval with local

\textsuperscript{5}http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/