INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Query Classification and Expansion in Just.Ask Question Answering System

Ricardo Miguel Pinheiro Pires

Dissertation for obtaining the Master's Degree in
Information Systems and Computer Engineering

Jury

President:	Professor Doutor Luís Eduardo Teixeira Rodrigues
Advisor:	Professora Doutora Maria Luísa da Silva Coheur
Evaluation Jury:	Professor Doutor Bruno Emanuel da Graça Martins

June 2012

Acknowledgements

I would like to thank my advisor, Professor Luísa Coheur for her guidance, discussion and motivation.

I want to thank all the members of L2F that provided me with essential tools for the development of this work and helped me with discussion. A special thanks to Ana Mendes for all the assistance provided and her endless availability.

Many thanks to my family for providing me with support each day and for always believing in me.

I want to thank all my colleagues that worked with me at Instituto Superior Técnico, with special mention to my friends André Matias, João Alves, Rui Correia, Teresa Gama, Carlos Perdigão, Mário Almeida, Sérgio Silva, Tiago Freitas, Guilherme Fernandes, José Lourenço and Andreia Guerreiro.

Last but not least, to my friends, Pedro Patrão, Ana Luísa, Pedro Carvalho, Filipa Reis, Leonardo Carneiro, Roberto Jacinto, Diana Conceição, André Beselga, Rui Almeida, Patricia Correia, Carla França, Tiago Santos, Liliana Lopes and Miguel Pinto for their care and support.

Lisboa, June 20, 2012

Ricardo Miguel Pinheiro Pires

To my beloved mother and father.

Resumo

O principal foco desta tese é o sistema de Pergunta Resposta Just.Ask desenvolvido no L2F, sendo os principais objectivos embutir o Just.Ask com mecanismos de Classificação de Queries e Reformulação de Questões/Expansão de Questões de forma a permitir aos utilizadores expressarem as suas necessidades de informação usando queries e melhorar o número de passagens relevantes obtidas, respectivamente.

Relativamente ao primeiro objectivo, um classificador de queries baseado em Support Vector Machines foi treinado e testado usando um conjunto de *features* desenvolvidas para a Classificação de Queries tendo em conta as características das queries, obtendo para a classificação geral e fina, respectivamente, uma precisão máxima de 86% e 80.2%. Adicionalmente, duas abordagens são propostas para realizar a Classificação Simultânea de Queries e Questões, tendo a melhor abordagem uma precisão máxima de 86.6% e 79.7% para a classificação geral e fina, respectivamente.

Quanto ao segundo objectivo, foram criados dois formuladores de queries que implementam, respectivamente, mecanismos de Reformulação de Queries e de Expansão de Queries. O primeiro formulador produz queries que representam possíveis reformulações da pergunta do utilizador que são obtidas fazendo *matching* da questão do utilizador com um conjunto de 163 expressões regulares. O segundo formulador expande a headword/headword composta e os verbos principais que existem na questão do utilizador com termos semanticamente relacionados obtidos através da Wordnet. Usando o Bing e o Lucene como motores de pesquisa observa-se, respectivamente, uma melhoria na performance inicial de 17 questões (9.2%) e 20 questões (10.9%) no número de questões com passagens positivas.

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.

Palavras Chave
Keywords

Palavras Chave

Query

Questão

Classificação de Queries

Classificador de Queries

Reformulação de Queries

Expansão de Queries

Keywords

Query

Question

Query Classification

Query Classifier

Query Reformulation

Query Expansion

Contents

List of Figures

List of Tables

List of Abbreviations

QA Question Answering

NLP Natural Language Processing

WWW World Wide Web

IR Information Retrieval

IQR Iterative Query Reformulation

IQE Iterative Query Expansion

AQE Automatic Query Expansion

GIR Geographical Information Retrieval

POS Part of Speech Tags

SVM Support Vector Machines

NYT New York Times

ODP Open Directory Project

POS Part of Speech

RBF Radial Basis Function

1 Introduction

In the last decade the Internet and the World Wide Web (WWW) have become invaluable resources for users worldwide to search information regarding a certain topic. Due to constant growing of the WWW, Web search engines were created in order to help users find specific information. Question Answering (QA) systems emerged in this context as a solution that provides users with the precise information that they seek. Instead of manually sifting through dozens of documents/web pages searching for a specific answer for a question, 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. For instance the question "How far is Yaroslavl from Moscow ?" could be done using three different queries: "distance Yaroslavl Moscow", "distance Yaroslavl to Moscow" or "distance Yaroslavl from Moscow". 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. Although both Question and Query Classification possess the same purpose, they are two distinct problems. Questions usually contain a lot of syntactic and semantic information that is crucial for accomplishing Question Classification. Queries, on the other hand, are characterized by the lack of this information which makes Query Classification an even more complex task when compared to Question Classification.

To obtain the exact answer to a question, QA systems (including Just.Ask) use Information Re-

trieval (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 is a solution to improve the number of relevant passages retrieved by a QA system. Query reformulation can be defined as "an attempt to improve poor users queries by: removing terms that deteriorate the retrieval performance, adding terms that aid retrieval, reweighting existing or new query terms to give the query a different emphasis, or a combination of those methods"(Billerbeck & Zobel, 2004). 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" (Billerbeck & Zobel, 2004).

1.1 Goals

The goals of the presented work are to create Query Classification and Query Reformulation/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 (Silva et al., 2010). 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. This goal raises two fundamental questions: "How is the classification going to be modeled?" and "How will the questions in the test datasets be transformed into queries that accurately represent the queries that a user would do for those questions?". There are also other important questions to this subject such as: "Which features used in Question Classification can also be used to accurately classify queries?"; "What kind of new features have to be considered for Query Classification?" and "How can simultaneous Query and Question Classification be accomplished?". A research of the literature regarding Query Classification is conducted in this thesis, in order to answer all aforementioned questions.

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. Both researches allow a better assessment of the advantages/disadvantages of each query expansion technique, and therefore allow an informed choice of which techniques to be added to the Query Formulator module.

1.2 Structure of this Document

The present thesis consists of six chapters, structured as follows:

- Chapter 2 describes the Query Classification and the Query Reformulation/Query Expansion state of the art, presenting the several advances achieved by the research community on these subjects throughout the years;
- Chapter 3 presents Just Ask's underlying architecture as well as the obtained results that are relevant for the Query Classification and Query Reformulation/Query Expansion tasks;
- Chapter 4 describes the main necessary modifications to Just.Ask architecture to support the task of Query Classification. More specifically, this chapter describes the created methods for converting questions in the training and tests sets into queries, the features developed for the Query Classification task, as well as the obtained results in the evaluation of that same task. Still related to the Query Classification topic, this chapter also depicts the created approaches to perform simultaneous Question and Query Classification, as well as the results obtained in the evaluation of this task;
- Chapter 5 depicts the devised Query Reformulation/Query Expansion approaches for Just.Ask's Query Formulator Module, as well as the results obtained by each approach;
- Finally, Chapter 6 presents the conclusions of this work, main contributions and what can be done in future work.

2 State of the Art

This chapter is divided into two main sections. 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 section 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.

2.1 Query Classification

Query Classification can be seen as multiclass categorization problem, which has been exhaustively studied in the machine learning literature. If a query is regarded as a short text segment, text categorization techniques can be applied to assign a category to that query. Text categorization uses as standard feature representation – the *bag-of-words* representation – where the word counts in the text document are used as features. In text categorization, each category is associated with a number of words that are representative of that category. Since text documents often contain a very large number of words, a number of representative words of each category will likely appear. Because of this fact, it is relatively easy for a standard machine learning method to find these words (even with a small amount of training data), and a weighted average of the words will provide a good estimate of which category a document belongs to.

This scenario changes drastically when the text in consideration is a query. Each query often contains only a small number of words, and thus it is very difficult for standard machine learning methods to find many representative words for a category from the training queries. The *data-sparsity problem* - many words that appear in the test queries do not occur in the training queries - is also likely to occur during all this process. All these facts corroborate the fact that the text categorization approach using the *bag-of-words* representation is not suitable for query classification.

Since the main obstacle in query classification is to deal with the lack of information inherent in a query, 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 trough query expansion based on initial search results (Section 2.2.3.2) or using knowledge structures (Section 2.2.3.1). Other approaches were developed with this same goal. For instance, Gravano and his team (Gravano

et al., 2003) classified queries with respect to their geographic locality in order to determine their intent - *global* or *local*. For example, the best results for the query "wildflowers" are probably web pages with *global* information about wildflowers, such as the types of climates wildflowers grow in, where wildflowers can be purchased, or what types of wildflower species exist. However, when a user that lives in Texas issues that same query, he may only be interested in retrieving web pages with information regarding existing wildflowers in Texas, i.e, web pages with *local* information. If the user's intent when submitting a query can be identified, the query can be modified by adding the most likely target geographical location, and therefore more suitable results are obtained from search engines.

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. Most of the contestants used the Web to enrich the queries and provide more context to the classification. The KDD-Cup task presented to the contents two additional challenges: translate the query classifier classifications into the target taxonomy and determine the query class based on document classifications. Solutions submitted by the participants have reasonable differences in their approaches. However, in an abstract level, all solutions consisted of three stages: preprocessing the set of labeled queries, gathering more information to augment these queries and modeling the query classification process.

For preprocessing the labeled queries, standard text mining techniques were used such as stopwords filtering, stemming and term frequency filtering, as well as more advanced techniques such as compound word breaking, abbreviation expansion and named entity recognition.

Several strategies were used to gather more information to augment the labeled query terms. Some participants used search results snippets, bag-of-words, search engine directories, search results titles or terms obtained using the Wordnet.

Regarding the query classification modeling, some participants choose to map the predefined/existing directory structure to the KDD-Cup categories, while others choose to construct mappings between KDD-Cup categories and bag-of-words of search queries, and then use the mappings to answer the categories of those search queries. For the modeling itself most of the participants adopted in their solutions, machine learning methods such as Support Vector Machines (SVM), K-Nearest Neighbors, Neural Networks, Logistic Regression and Naive Bayes. Among these models, queries and other relevant knowledge were represented by n-grams or by the feature-value representation.

The winning solution of the KDD-Cup (Shen et al., 2005) uses an ensemble[1] of classifiers in conjunction with multiple search engines. More specifically, each query to be classified is enriched by its related

[1] An ensemble is a set of models whose predictions are combined by weighted averaging or voting.

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.

The synonym-based classifier uses the category information associated with the Web pages - search engine category - that is collected for each query trough different search engines. However, search engine category taxonomies vary according to each search engine and they are also very distinct from the taxonomy provided by the KDD-Cup for the classification. To map between search engine categories and KDD-Cup categories, a keyword matching is performed. If a search engine category and a KDD-Cup category share some of the same keywords, then a direct mapping can be performed.

The statistical-based classifier uses SVM, where the contents of the related pages of each query are used as classification features. The training data for the classifier is obtained by leveraging the mapping function devised for the synonym-classifier and the manually labeled Web page directory Open Directory Project. The mapping function is utilized to find, for each KDD-Cup category, a set of categories from the ODP directory. Web pages contained in those collected ODP categories are then used as the training documents for the target KDD-Cup category.

Finally, 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 Reformulation and Query Expansion

A user query is only one of the many possible ways to state the information that the user seeks. So there is often a discrepancy between the terminology utilized by the user, and the terminology used in the documents that fulfill the user information need. This discrepancy causes a substantial reduction in the documents retrieved by the system since the documents that do not contain any keywords of the user query are not retrieved. Query Reformulation and Query Expansion techniques are a solution to this problem.

The organization of this section is as follows: first, Sections 2.2.1, 2.2.2 and 2.2.3 describe the most important query reformulation/expansion techniques that have been studied throughout the years. Some of the query expansion (and reformulation) techniques that are used in existing QA systems are discussed in Section 2.2.4.

2.2.1 Iterative Query Reformulation

This main idea of Iterative Query Reformulation (IQR) is to use in an iterative fashion feedback from the user, to improve the quality of the query by adding/removing query terms or by reweighting the existing query terms. *Relevance Feedback* is probably the most popular IQR technique in IR since a great amount of research was conducted into this technique throughout the years (Ruthven & Lalmas, 2003). Relevance feedback works as follows: an initial search with a user query is performed and set of documents is retrieved. Then, from short text snippets (usually of titles or abstracts of the retrieved documents) the user identifies the set of relevant documents. Those documents are then used to modify the query by reweighting the existing query terms and/or by adding/removing terms that aid/deteriorate the retrieval performance. The goal of this whole process is to modify the original query so that the new query becomes more similar to the relevant documents. Relevance feedback, can be implemented in various ways depending on the methods used to select terms for the feedback query (some of this methods can be found in (Efthimiadis, 1993, 1995)) and also on the techniques used to rank the documents. Vector space, probabilistic and language models are commonly document ranking techniques used in both query reformulation and query expansion techniques.

In vector space models, both queries and documents are represented as vectors in a $|V|$-dimensional vector space where V is the set of all distinct terms in the document collection. Documents are classified as relevant or non-relevant according to the similarity between them and the query vector. The user judgements of the relevance of some of the classified documents are used as examples for updating the query vector. Rocchio's algorithm (Rocchio, 1971) is the standard algorithm for relevance feedback using a vector space model. The algorithm's main objective is to find an "optimal" query vector that maximizes the similarity with relevant documents, while minimizing the similarity with non-relevant documents. The following formula calculates (trough various iterations) that "optimal" query vector:

$$\overrightarrow{q_m} = \alpha \overrightarrow{q_0} + \beta \frac{1}{|D_r|} \sum_{\overrightarrow{d_j} \in D_r} \overrightarrow{d_j} - \gamma \frac{1}{D_{nr}} \sum_{\overrightarrow{d_j} \in D_{nr}} \overrightarrow{d_j}$$

The variable q_m denotes the modified query vector, q_0 the original query vector, D_r and D_{nr} the sets of known relevant and non-relevant documents respectively and finally the variables α, β and γ are the original query, relevant documents and non-relevant documents weights respectively.

A probabilistic model ranks documents according to their evaluated probability of relevance to the user query. Probabilistic models are usually grouped in two different categories: relevance models and inference models. Relevance models determine the relevance of documents using statistical estimates of that relevance. The main problem in estimating the probability of relevance for every document of a collection is the large number of variables involved in the representation of documents, and the small

amount of existing information regarding the relevance of those documents. The existing relevance models differ in the way how documents and queries are represented, and how the estimated probability of a document to a user query is calculated. Inference models, unlike relevance models have a more complex form to evaluate relevance. With inference models information not present in the user query (such as domain knowledge, knowledge about the user, user relevance feedback) is also used in the evaluation of the relevance of a document. This characteristic causes inference models to be less collection-dependent when compared to relevance models, since parameters in relevance models used for estimating the probability of relevance of a document are valid only for the current collection. A description of the main relevance and inference models can be found in (Crestani et al., 1998).

Unlike the other approaches, where documents are simply perceived as a conjunction of several strings of terms put by an author, in language models documents are seen as one of the possible formulations of the information that the author tried to convey. The rationale for this approach is very simple: different authors use different terms or different sentence structures to convey the same information. Language models are usually built from term distributions of a document at hand, and from term distributions of a large collection of documents, due to the usual lack of information about the language model of the document at hand. The actual ranking of the documents is based on the relationships between the query and documents language models, and three different approaches can be used to accomplish that goal. In the first approach documents are ranked against a query by the probability that the query model was generated by a document model. The second approach is the complete opposite of the first, so probabilities are assigned to documents according to their likelihood of being generated by the model of a query. Finally, the third approach performs a comparison between the query and the model of each document. Some papers describing language models modeling can be consulted in (Croft & Lafferty, 2003).

2.2.2 Iterative Query Expansion

Iterative Query Expansion (IQE) is similar to iterative query reformulation, in the sense that both try to improve the quality of the user query trough various iterations. However, as opposed to IQR methods, IQE only improves the query by adding new terms. So, in each iteration the user is presented with a list of potential expansion terms obtained from a system run with the last improved query. There are several important factors in this step, such as the number of expansion terms that the user must choose, the number of expansion terms presented to the user and the criteria that should be used to select potential expansion terms. After the selection of a set of expansion terms, these terms are weighted according to a certain criteria (for example according to how many of the relevant documents the terms occurred in) and are incorporated in the user query.

2.2.3 Automatic Query Expansion

This main idea of Automatic Query Expansion (AQE) is similar to the iterative query expansion with one major difference: in AQE methods the user does not give any explicit feedback to the system regarding the terms to be selected for expansion. Since in AQE the systems provide themselves with relevant feedback without any user intervention, AQE is also known as *pseudo relevance feedback* or *automatic relevance feedback*. AQE uses for the expansion of query terms three different strategies that will be discussed in the following sections.

2.2.3.1 Expansion Trough Knowledge Structures

In this approach, expansion terms are collected from previously developed knowledge structures that contain information about a set of terms (synonyms, related terms, etc.). These knowledge structures are usually term dependence matrices, lookup tables or graphs and can be divided into two classes: collection independent and collection dependent knowledge structures.

Collection independent knowledge structures, are built without the use of statistics extracted from the document collection/corpus used by the QA system. Due to this fact, query expansion algorithms based on these knowledge structures are also known as *external techniques*. Some examples of collection independent knowledge structures are: domain specific *thesaurus*[2], general-purpose thesaurus and lexicons and dictionaries (such as the Collins dictionary). Wordnet (Fellbaum, 1998) is a general-purpose thesaurus that deserves a special mention. Several studies (Voorhees, 1994; Hsu et al., 2008; Zhang et al., 2009) show that Wordnet can be used to expand (and consecutively improve) individual queries.

Collection dependent knowledge structures in contrast to the previous ones, are built upon statistics of the document collection/corpus used by the QA system. Query expansion algorithms based on collection dependent structures are also referred as *global analysis techniques*. Global analysis techniques assume that all words (except stop-words) in a query are *concepts* and that the global context of a *concept* can be used to determine similarities between concepts. An important example of global analysis techniques is latent semantic indexing (Deerwester et al., 1990), which builds on both term co-occurrence[3] and term clustering[4]. Latent semantic indexing works as follows: first, an index of the collection terms is created after removing adjectives and terms that have multiple occurrences across several documents (or in a single document). Then, terms are organized in a multidimensional space according to the following criteria: terms that are close to each other in the collection, i.e, terms that co-occur at document,

[2]In this context a thesaurus is a manual construction (usually performed by experts on the domain) of a list of terms, their respective synonyms and other related concepts to those terms.

[3]The tendency of two terms to co-occur in similar contexts can be seen as an indicator of semantic similarity between those terms. Usual contexts for term co-occurrence are appearing in the same document, in the same sentence or following the same word.

[4]Terms are grouped together along any pre-defined criteria.

sentence or word levels, are also closely located to each other in the multidimensional space. Finally, terms in the query can be expanded by using its respective closely related terms. The distance between each expansion term and the original query term conveniently gives an indication of an appropriate weighting of each expansion term.

The main problem with expanding queries trough knowledge structures, is that only the context of individual query terms is used for expanding the query and a term can take on different meanings, depending on the context that it appears in.

2.2.3.2 Expansion Based on Initial Search Results

To solve the problem described in the last section, a initial set of results using the original query can be obtained and incorporated into the query. More specifically, a set of terms are selected from the top N-ranked documents based in the frequency of those terms in those top-N ranked documents as well as collection statistics. From this set of terms, the terms with the highest ranks are added to the query (with a weight according to their respective rank). As opposed to global analysis, where only the individual context of query terms are captured, this expansion technique makes a better disambiguation of the several meanings that a term might have. This disambiguation is a consequence of terms being extracted from documents that are retrieved considering the whole query, rather than individual query terms. For this reason, expansion based on initial search results is also known as *local analysis*.

Although local analysis performs better than expansion trough global analysis (J. Xu & Croft, 2000), it still has some drawbacks. One drawback of this method is the low efficiency (when comparing with global analysis methods), since the terms occurring in the local set of documents need to be assessed through a series of expensive disk accesses. Billerbeck (Billerbeck & Zobel, 2004) showed that local analysis expansion via document surrogates (tf.idf summaries were used as documents surrogates) can improve that efficiency. However, the main hindrance of local analysis is that a whole document is used for selecting expansion terms and may cover several different topics, which in turn results in irrelevant information for the query at hand. The existence of multiple topics and irrelevant information leads to a noisy feedback model, as terms from non-relevant topics may be picked for expanding the query. In the worst scenario the use of non-relevant terms for expansion can cause the alteration of the focus of the query - problem known as *query drift*. So it is of the utmost importance to select from relevant feedback document terms that are relevant to the query topic. From the several strategies that have been devised with this purpose a few deserve a special mention.

Yuanhua Lv et al. (Lv et al., 2011) propose a learning algorithm - FeedbackBoost - that iteratively selects and combines basic feedback methods to improve the overall effectiveness and robustness of the relevance feedback. The motivation for this approach is simple - basic feedback methods with different strategies for document weighting produce different weights for different query topics and therefore are

complementary to each other. Experiments performed by the authors showed that FeedbackBoost improved average precision and reduces the number and magnitude of feedback failures when compared to: feedback methods based on language models (the SMM2 mixture model and the RM3 relevance model), REXP-FB method (a recent work on improving robustness of pseudo feedback) and AdaRank (an existing learning to rank approach that can also improve both robustness and effectiveness as much as FeedbackBoost does).

Yang Xu et al. (Y. Xu et al., 2009) suggest a query-dependent method for pseudo relevance feedback using existing information at Wikipedia. Wikpedia articles are constantly being created/updated and are semi-structured, characteristics that make Wikipedia an ideal information source for query expansion. Regarding the method itself, first queries are categorized into three types (entity, ambiguous and broader queries) and relevant documents are obtained from the top N-ranked documents retrieved in response to the query or using Wikipedia entity pages corresponding to queries. The expansion terms are then selected from the documents taking into account the term distributions and the structure of Wikipedia pages. This process is performed by two methods. The first method replaces the term frequency of a document (previously calculated with a relevance model) with a linearly combined weighted term frequencies. The second method is a supervised learning method (SVM to be more precise) in which features derived from Wikipedia fields are used to train the classifier. The authors demonstrated that their query-dependent method has a larger medium average precision than a query-likelihood language model used as baseline for the experiments.

2.2.3.3 Query Expansion with Associated Queries

Global and local analysis expansion techniques obtain the expansion terms from one collection of documents. Expansion with associated queries differs from these two since it can use for that purpose, any document collection that present a similar topic coverage to the documents present in the "original" collection. This similar document collection can be created using *query association*. For each document in the original collection, a surrogate is built consisting of past queries that are associated with that document (in order words, queries that share a high statistical similarity with the document).

Regarding the association process, a query is submitted to a search system, and a similarity score is calculated based on the similarity of the query at hand to documents that contain query terms. Then, the query becomes associated with the top N-ranked documents that held the highest similarity scores. Each document have a maximum number of query associations, to assure a certain level of similarity to the queries that are associated to him. This way, when the maximum number of associations is reached the least similar query can be replaced with a more similar query.

The expansion process is performed in two phases - the ranking and selection phases. The ranking phase consists in submitting a query to a search system and retrieving the top-N ranked documents

(from a particular collection or the surrogates corresponding to those documents). From these top-N ranked documents (or the surrogates corresponding to those documents), a certain number of expansion terms are selected and appended to the original query. Billerbeck and Zobel (Billerbeck & Zobel, 2004) showed that this query expansion approach lead to better results relatively to expansion based on initial search results.

2.2.4 Query Reformulation and Query Expansion in Existing QA Systems

It is important to clarify that throughout the rest of this work we will denote as "query reformulation" what is called as "query formulation" in typical ad-hoc IR systems. The motivation for this decision comes from the fact that all descriptions of "query formulation techniques" in the literature of QA systems are referred as "query reformulation techniques", so we chose to use the same terminology in order to maintain the consistency with the QA literature.

The following sections describe the query expansion and reformulation techniques that are used in four different QA systems. These systems present the techniques that we found more interesting during our research, and that are worth analyzing in a deeper level. Other factors also motivated this decision. The fact that Open Ephyra and Aranea are open source systems is one of those factors. If some on the techniques used in these two systems (or similar approaches to them) were to be implemented in Just.Ask, consulting the source code of these systems can be a great help in accomplishing that goal. Another motivation for choosing Aranea was the fact that as opposed to the other systems described in this section (that use solely query expansion or query and reformulation techniques), Aranea only uses reformulation techniques. The improvements obtained in Raposa and Quercol systems with the use or their expansion techniques are also another important factor for choosing to describe those techniques of the two systems.

During our research many other QA systems were analyzed such as: AnswerBus (Zheng, 2002), AskMi (Sakai et al., 2004), Maya (Kim et al., 2001) and Qarab (Hammo et al., 2002). These systems were not described due to the lack of information regarding query reformulation/expansion techniques utilized or the use of techniques already included in the systems referred in the previous paragraph.

2.2.4.1 Open Ephyra System

Open Ephyra (Schlaefer, Gieselmann, et al., 2006; Schlaefer, Gieselman, & Sautter, 2006; Schlaefer et al., 2007) utilizes 5 types of queries: keyword, term, predicate, reformulation and interpretation queries. All these types of queries are generated based on two representations of the user questions. Interpretation queries use the first representation, where all verbs are replaced by their infinity form and all nouns are reduced to the singular form. The remain query types use the other representation, where the verb

constructions are replaced by the verb construction that is expected to appear in the answer, e.g, "did ... occur" is replaced by "occurred". All descriptions bellow (except the interpretation queries description) will provide the correspondent query formulations to the question "In what year was the CMU campus at the west coast established?".

Keyword queries as the name indicates are queries based on the most important content words of the question. For our example the keyword query would be "CMU campus west coast established".

Term queries consist of the question terms (single tokens or expressions) expanded with semantically related concepts. WordNet is used, following an approach similar to (Sun et al., 2005), to get the semantically similar terms that are used as expand terms. For the example the term query would be ''(CMU OR ''Carnegie Mellon") campus ''west coast" (established OR founded OR launched)".

Predicate queries are formed from the predicate verb and its arguments. For the example the corresponding predicate query is "the CMU campus at the west coast established".

Reformulation queries consist of reformulating the question into a sentence (answer-pattern) where the answer to the question is likely to be found, and be extracted directly. For the example the corresponding reformulation query would be "CMU campus at the west coast was established in".

Interpretation queries are based in what the authors of the system call of "question interpretation". The authors consider that all questions can be reduced to three components - A question asks for a *property* of a *target* in a specific *context*. These components are extracted using a set of manually defined patterns and a query string is formed with the concatenation of the target object, all context objects and additional keywords from the question. For example the question "How many calories have a Big Mac?", where *Property*$\rightarrow$ NUMBER, *Target* $\rightarrow$ calories and *Context* $\rightarrow$ Big Mac has as query the string "calories Big Mac".

2.2.4.2 Quercol System

Quercol (Cardoso et al., 2007) is a Geographical Information Retrieval (GIR) system developed for the GeoCLEF task. GIR systems are concerned with the retrieval of information from documents that contain spatial (geographic) references expressed in multiple languages (that may not be the same as the question language). GeoCLEF evaluates GIR systems for search tasks involving both spatial and multilingual aspects. Quercol uses pseudo relevance feedback as query expansion approach and utilizes queries in the Disjunctive Normal Format - a query is represented as a disjunction of conjunctive terms (both original and extended terms).

Initially for each user topic[5], stop-word terms from the title are extracted and the remaining terms

[5]A topic refers to a question describing a spatial user need.

are combined into a boolean query with AND operators. This query is submitted to the system in order to obtain a set of documents that will be classified as relevant or non-relevant. Based on the corpus at hand, Quercol defines a threshold value to which documents above it are considered as relevant and the remaining documents are considered non-relevant. Relevant documents are then tokenized and their terms are weighted (using the $w_t(p_t\text{-}q_t)$ algorithm (Efthimiadis, 1993)), creating a ranked term list that represents the most important information contained by the relevant documents. Finally, the queries in the Disjunctive Normal Format are constructed using terms from the ranked term list and from the topic title. These terms can be combined using two different approaches: the Logical combination and the Relaxed combination.

The Logical combination considers that all non-stop words from the user topic represent *concepts* that are mentioned in the retrieved documents, and that each query must contain at least one term for each concept in the user topic. The terms for each concept are extracted from the ranked term list if they have the stem as the corresponding concept, and a *concept bag* with those concepts is created. After the creation of all concepts bags, the remaining top terms from the ranked term list are placed in a *expanded terms bag*. Finally, all combinations between *m* bags and *n* terms in each bag are generated producing *m* x *n partial queries*, i.e, each *partial query* contains one term from each bag. With the exception of terms from the expanded terms bag that are joined by the OR operator, all terms are joined by the AND operator resulting in the following query format:

$$OR\{OR\{\textit{partial queries}AND\{conceptbags\}\}, OR\{\textit{partial queries}AND\{conceptbags{+}expandedtermsbag\}\}\}$$

The Relaxed combination is an improvement of the Logical combination, since this approach forces all concepts to appear in the query strings, which can lead to not retrieve several relevant documents. To solve this problem, the Relaxed combination only uses a single bag - *single concept bag* - to collect the related terms instead of several concept bags. All combinations of the two bags (*single concept bag* and *expanded terms bag*) and the *n* terms in each bag produce 2*n* partial queries that are joined by the AND operator. The final query format in this approach is:

$$OR\{partialqueries AND\{singleconceptbag + expandedtermsbag\}\}$$

2.2.4.3 Raposa System

Raposa (Sarmento, 2008) is a system that responds to factoid questions in Portuguese. Raposa uses three types of queries: keyword, pseudo-stemming and verb expansion queries.

Keyword queries are just simple queries with the most important content words of the question.

Pseudo-stemming queries are a simple improvement of keyword queries. In this query type, terms

that are not identified as named entities have their suffixes substituted by wild-cards, in order to generalize the query so that more text snippets can be retrieved. At the context at hand, suffixes were considered to be the last 2-4 characters of terms with more than 5 characters long.

Verb expansion queries expand the main verb of the question to semantically equivalent or related verbs, and are used for answering factoid questions related to actions or events like "Who killed J.F.K?". In these type of questions the verbs have an important role in retrieving relevant text snippets and, by "expanding" them, more relevant text snippets can be retrieved. Due to the lack of available lexical tools like the Wordnet for Portuguese, to obtain the desired verb expansions, a verb thesaurus was build using statistical processing of a large corpus. The full description of the approach used to automatically generate a verb thesaurus from a corpus can be consulted in (Sarmento, 2008). Verb expansion queries are constructed in the following fashion: first, the verb is identified and its corresponding radical is found. Using the verb thesaurus the top five related verbs to the radical are found. Then, pseudo-stemming is applied to the radical and related verbs by substituting the last character by a wildcard (in order to retrieve more text snippets). Finally, the radical and expansions are joined by an OR operator. For the example above one possible verb expansion query would be "Who (OR kill* OR shot OR assasinat* OR murder*) J.F.K".

2.2.4.4 Aranea System

Aranea (Lin, 2007) is a system that responds to factoid questions using two different approaches. One searches answers in structured resources and the other uses data redundancy of unstructured information sources (like the Web) to do so. In the latter, Aranea uses three types of queries: baseline queries (the question in natural language submitted by the user), exact queries and inexact queries. An exact query is a reformulation of the baseline query that anticipates the specific location of the answer (in the retrieved passages) and tries to extract it. For instance, the answer to the question "Where is The Grand Canyon?" will probably appear next to the phrase "The Grand Canyon is located in". Therefore, the correspondent exact query of that question is "The Grand Canyon is located in ?y", where ?y indicates the location of the anticipated answer. Exact queries are generated using manual pattern matching rules based on question terms and their POS tags. Each time a pattern is matched at the morpho-lexical level, the correspondent exact query is generated. Exact query formulation patterns range from general patterns like "What did A verb B $\rightarrow$ A verb+ed B ?y", to particular question types like the one presented in the example. Formulation patterns can manipulate the sequence of tokens and their morphology, but they can also insert additional words that were not present in the original question like seen in the example.

An inexact query is also a reformulation of the baseline query and is generated by the same pattern matching rules of exact queries. This query type is similar to the reformulation queries of Open Ephyra,

and presents the benefit of a broader coverage since an exact match of the inexact query is not required. The inexact query for the above example would be “The Grand Canyon is located”.

3 Just.Ask Overview and Results

Just.Ask (Silva et al., n.d.) possesses the standardized QA architecture (Jurafsky & Martin, 2008) (depicted in Figure 1), consisting of a pipeline with three main modules: Question Interpretation, Passage Retrieval and Answer Extraction. These modules will be described in the subsequent sections.

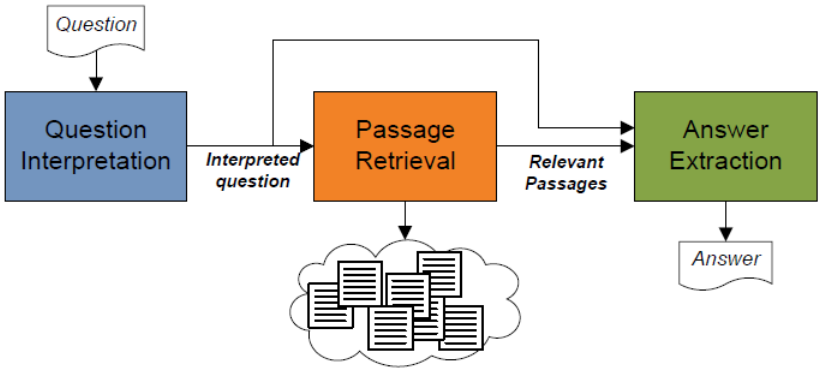

Figure 3.1: Just.Ask Architecture.

3.1 Question Interpretation

The Question Interpretation module (presented in Figure 3.2) is responsible for understanding the question. It receives as input a question from the user, expressed in natural language and outputs an interpreted question. The interpreted question includes the original user question and useful information for the Passage Retrieval module (such as the various tokens of the question and their correspondent POS tags, the question headword and headword synonyms, the compounded headword and the question category). This module performs two main tasks: question analysis and question classification.

This work's contributions will enable Just.Ask to also receive a user query as input. In this case, the module output will be similar to the question's output, namely, an interpreted query that includes the original user query and useful information the Passage Retrieval module (query tokens and their correspondent POS tags, and the query category). The previous mentioned tasks will be adapted to address this new input type.

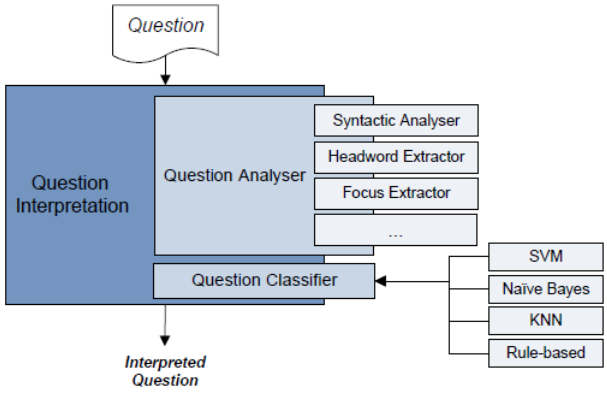

Figure 3.2: Detailed view of the Question Interpretation module in Just.Ask.

3.1.1 Question Analysis

The goal of question analysis is to gather useful information about the question for the Passage Retrieval module and to provide the tools for extracting the features used for Question Classification. For this purpose several NLP tools are used. A tokenizer is used to identify the various tokens that are in the question. The parse tree of the question (tree with the syntactic structure of the question) is obtained using the Berkeley Parser (Petrov & Klein, 2007) trained on the QuestionBank (Judge et al., 2006). The POS tags (which represent the grammatical classes of the question tokens) are also obtained by this method.

As for the question headword (word with the information that is being sought after), its extraction method consists of traversing the parse tree of the question, top-down, using a set of predefined rules – the head rules. These are based in the rules presented in (Collins, 1999), however, they were modified to extract headwords from questions. For cases where the head rules cannot successfully extract the headword, another set of rules – non-trivial rules – are used for that purpose. The headword category is obtained using a set of hand-written rules to map the headword to a category.

Regarding the compound headword (combination of the question headword with more words that constitute a single unit of meaning), its method of extraction consist of identifying a compound word that includes the headword of the question. Both headword and compound headword help to classify the question into a finer grained category.

For questions without headword which are harder to classify properly, a set of lexical and syntactic patterns are used to assign a category to these questions. These same lexical and syntactic patterns are used in two other different contexts. They are utilized to assign a category to questions that have explicit information about the expected answer type. For instance, the question "How old is the Empire State Building?" contains the word "old" which indicates that the answer must be a number (in the form of

numeric digits or written number). These lexical and syntactic patterns are also utilized to determine the focus of non-factoid questions and questions about acronyms or counts.

3.1.2 Question Classification

Question classification purpose is to assign a predefined semantic category that represents the expected semantic type of the answer. It is a step of the utmost importance, since the semantic category can be used by the Passage Retrieval Module to retrieve only those passages that contain at least one occurrence of the expected question type, therefore reducing the number of candidate answers.

Just.Ask uses the Li and Roth (Li & Roth, 2002) two-layer question taxonomy for question classification, which consists of six coarse grained categories that are further divided into fifty finer grained categories. This taxonomy is one of the most widely used by researchers in QA due to the fact that the authors have published a set of 5500 labeled questions that are freely available on the Web. The use of this taxonomy allows the result comparison with many of the existing QA systems.

The main challenge in question classification is to identify the set of features that lead to an accurate classification. With this purpose it is important to understand how each feature contribute in isolation and combined with other features. Just.Ask uses three types of features: lexical, morpho-syntactic and semantic features. These features are extracted by the existing tools in the Question Analysis module.

As for lexical features, *n*-grams, binary *n*-grams and stems are used. A *n*-gram is a sequence of n consecutive words from a question. Its use is motivated by the fact that questions of the same category usually share *n*-grams. For instance LOCATION:MOUNTAIN type of questions usually contain the unigram *mountain*. Just.Ask uses Unigrams, Bigrams and Trigrams. To illustrate the difference between *n*-grams and binary *n*-grams let us consider the question "Who is the richest person in the world ?". Using *n*-grams the unigram "the" has a value 2, i.e, indicates two occurrences of the word "the" in the question. On the other hand, using binary *n*-grams the unigram "the" has as value 1, i.e, the unigram "the" occurs in the question. Therefore, binary *n*-grams represent the presence (or absence) of a *n*-gram in the question trough a binary value (0 or 1).

A stem is the grammatical root of a certain word. Stemming is, therefore, a technique that reduces words to their correspondent stems (for example, after stemming "playing" and "played" would become play). Just.Ask uses stemming after representing the question using *n*-grams to transform each word of a *n*-gram into its corresponding stem. Stemming is used in combination with the removal of stop words, to reduce the total number of features that have to be considered by the classifier.

Regarding morpho-syntactic features, POS tags and the question headword are used. The process of extracting and using these features was already described in the previous section. Named entities and semantic headword are used as the main semantic features.

Named entity recognition consists of finding and classifying particular names in text into predefined categories.

After the extraction, the recognized named entities can be added as features to the classifier - feature Ner-Increment - or can replace the identified named entities with the correspondent entity category - feature Ner-Replace. Just.Ask uses three different approaches provided by the LingPipe library[1] to extract named entities from questions. The first approach performs a match of the question with a set of regular expressions which enable the recognition of named entities when a match between a question and a regular expression occurs. The second approach extracts named entities by performing a match between the question tokens and lists of names (and aliases) for entities and their types. When a match between a question token a name occurs, the entity type of the name is assigned to the query token. Finally, the last approach consists in using a Hidden Markov Model based entity recognizer trained for questions. After the extraction, the recognized named entities can be added as features to the classifier - feature Ner-Increment - or can replace the identified named entities with the correspondent entity category - feature Ner-Replace.

The semantic headword improves the headword feature by enriching it semantics. Often headwords of questions have a semantic field that is included within the semantic field of another word (which is called the hypernym). For instance, the questions "What pilot has the nickname of "Iceman"?" and "What actor first portrayed Batman?" have pilot and actor as their headwords, respectively. Both actor and pilot are hyponyms (specializations) of the word *person*, thus *person* is a hypernym of actor and pilot. The awareness of this fact would allow to classify both questions with the HUMAN:INDIVIDUAL category.

To exploit the relationship between hyponyms and hypernyms Just.Ask utilizes Wordnet lexical hierarchy to associate a higher-level semantic concept to a question headword. For that purpose, sets of related Wordnet synsets are grouped into fifty clusters (each cluster represents a fine-grained category). The question headword is then mapped into a Wordnet synset using a set of heuristics. Finally, a breadth-first search is performed on the translated synset's hypernym tree in order to find a synset that belongs to any of the pre-defined clusters.

As for the classification itself, it can be modeled according to two types of techniques: hand-built rules or machine learning techniques like SVM and Naive Bayes.

3.1.3 Question Classification Results

To evaluate the accuracy of the question classifier with the features mentioned in the previous section, the SVM model is utilized. Support Vector Machines are machine learning algorithm used for classifica-

[1]Software available at http://alias-i.com/lingpipe.

tion and regression analysis. SVM construct a N-dimensional decision boundary surface (often called a hyperplane) that optimally separates data into positive examples and negative examples, by maximizing the margin of separation between these examples. SVM can also perform multi-class classification, i.e, perform the classification of several classes. The dominant method for doing so is to reduce the single multi-class problem into multiple binary classification problems, by building regular SVM binary classifiers which distinguish between one of the classes and the rest (one-versus-all approach). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier that classifies that test data with the greatest margin is chosen.

Specifically, Just.Ask uses the LIBSVM[2] implementation with a linear kernel is used and the classifier is trained using the one-versus-all-multi-class approach. A set of 5500 questions are used to train the SVM model and a set of 500 questions are used for testing. Both the training and test sets are classified following the Li and Roth taxonomy.

When only using lexical features, the combination of unigrams and bigrams held the most accurate results having an accuracy of 90.4% and 81.2% for coarse- and fine-grained classification, respectively. As for the solely use of syntactic features the question headword obtained the best results with an accuracy of 92.6% and 82.8% accuracy for coarse- and fine-grained classification, respectively.

The combination of the previously lexical and syntactic features provided interesting results. For example, for fine-grained classification unigrams and the question headword resulted in 85.4% of accuracy. However when combining unigrams, bigrams and the question headword this accuracy dropped to 83.6%. This was not expected, as the combination of unigrams and bigrams produced the best result when considering only lexical features.

The combination of the question headword with unigrams is therefore, the most discriminative lexico-syntactic feature.

The different semantic features were combined with the lexico-syntactic features that held the best results, in order to analyze the contribution of the semantic features to question classification.

The combination of question headword, unigrams and semantic headword obtained the best result for coarse-classification with an accuracy of 95%, while the combination of question headword, binary-unigrams and semantic headword yield the best result for fine-grained classification with 90.4% of accuracy. As for named entities, the experiments showed that these help to improve the classification accuracy when combined with unigrams and question headword. A more detailed analysis of the impact of all this features in Question Classification can be found in (Silva et al., 2010).

[2] A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/simcjlin/libsvm.

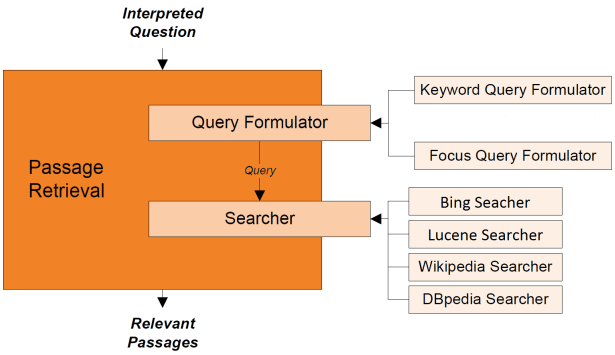

Figure 3.3: Detailed view of the Passage Retrieval module in Just.Ask.

3.2 Passage Retrieval

The Passage Retrieval module (depicted in Figure 3.3) receives an interpreted question from the Question Interpretation Module and outputs the relevant passages (passages which are more likely to contain an answer) for that question. Based on the question category, Just.Ask uses different information sources (like Bing and Wikipedia) and different query formulation techniques (according to the type of information source being used). The most simple query formulation strategy, the keyword-based strategy, consists in generating a query that has all the words in a given question, except stop-words, question words and punctuation. This query formulation strategy is used to generate queries for Bing and Lucene search engines to answer factoid-type questions. The focus query formulator, as the name states, builds a query uniquely with the focus of the question, and it is used to generate queries for Wikipedia and DBpedia, to answer non-factoid type questions - questions classified as DESCRIPTION_DEFINITION and HUMAN_DEFINITION.

3.2.1 Passage Retrieval Results

To evaluate the performance of the Passage Retrieval module (without the use of any query expansion techniques) the system was ran using a corpus with 183 factoid questions and two different copora for answers: the Web and paragraphs from the New York Times (NYT) 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, the Keyword Query Formulator is used to generate queries to those search engines. As a consequence of using the Web as corpus for answers, it is important to take into account that the results obtained have a certain inaccuracy due to the inherent changeability of the Web. However even with this associated inaccuracy, the obtained results can still be utilized in order to evaluate the module performance. In order to reduce this inaccuracy to a minimum, for each test using the Web as corpus, the system was run three times being the final result of that test the average of the results

obtained in all runs.

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. Using each one of the parameterizations in the previous paragraph, the system was run the defined maximum number of passages to retrieve per query, with the goal of analyzing the number of passages that contain at least one correct answer. These passages are usually denoted as *positive passages*.

Keyword Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	137 (74.9%)	0.37	0.5
Lucene	102 (55.7%)	0.19	0.33

Table 3.1: Passage Retrieval baseline results.

Table 3.1 presents the results of this evaluation for both Bing and Lucene search engines. More specifically, the table presents for each search engine the number of questions for each there is at least one positive passage (#Question$_{1+PosPassage}$), the mean reciprocal rank of the first positive passage for all questions (MRR$_{ALLQ}$) and the mean reciprocal rank of the first positive passage only for the questions for which at least one positive passage exists (MRR$_{QPOSPASSAGES}$).

3.3 Answer Selection

The Answer Selection module receives the interpreted question and the relevant passages from the Interpretation and Passage Retrieval Modules respectively, and returns the answer to the user question. This module performs two main tasks: candidate answer extraction and answer selection.

Candidate answer extraction uses different extraction techniques according to the question type taxonomy utilized in the question classification step. Regular expressions are utilized as an extraction method for NUMERIC type questions. A machine learning-based named entity recognizer is used for HUMAN:INDIVIDUAL type questions. There are questions like "*Which animal is the slowest?*", in which the answer is not an instance of animal but a type of animal. To extract candidate answers from this type-of questions Just.Ask uses WordNet's hyponymy relations, since the candidate answers for these questions are often hyponyms of the question headword. For LOCATION:COUNTRY and LOCATION:CITY type questions, a gazetteer (a geographical dictionary) is used to help extract the candidate answers.

Regarding answer selection, this process is done in three steps. The first step is Candidate Answer Normalization which consists of reducing to a canonical representation answers that belong to the categories NUMERIC:COUNT and NUMERIC:DATE. The second step, Clustering, consists of grouping similar answers together according to some similarity criteria (the overlap distance or the Levenshtein distance). Each cluster has a score assigned to it, in which in the most of the cases corresponds to number of members in the cluster. Also each cluster has an answer representative that is defined as the longest answer in the cluster. The final step is Filtering, which as the name seems to indicate, it consists in removing undesired answers according to a certain filtering criteria. In this case, if any of the answers present in any of the clusters is contained in the original question, than the whole cluster is discarded. After this filtering process the representative answer of the cluster with the highest score is returned

4 Query Classification

This chapter is devoted to the creation of Query Classification mechanisms in Just.Ask system, in order to enable users to express their information needs in form of queries. The chapter is divided into four sections that can be perceived as three coarse parts.

The first part, Section 4.1 describes the created methods for converting questions in the training and test datasets into queries, as well as an user evaluation of the same methods to ascertain which method produces queries that are more similar to the ones submitted by the users.

Sections 4.2 and 4.3 constitute the second part. These sections describe the features developed for the Query Classification task, the results obtained by the devised Query Classifier and an experiment to improve the results of the previous classifier.

Finally, the last part which corresponds to Section 4.4, depicts the created approaches to perform simultaneous Question and Query Classification, as well as the results obtained in the evaluation of this task.

4.1 Adapting the training and test datasets

As it was mentioned in Section 3.1.3, Just.Ask models Question Classification using a SVM model with a linear kernel and using the one-versus-all-multi-class-approach. This model is trained and tested with a dataset of 5500 and 500 questions respectively, classified according to the Li and Roth taxonomy. The Query Classification task is performed using the same training and testing datasets with the same setting of Question Classification due to the reasons depicted in the beginning of Section 4.2. The idea of using existing training and testing query datasets was discarded since most of these datasets possess queries that do not represent queries that a user would do to a QA system.

In order to use the Query Classification 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. For instance, users could do the question “How far is Yaroslavl from Moscow ?” using three different queries: “distance Yaroslavl Moscow”, “distance Yaroslavl to Moscow” or “distance Yaroslavl from Moscow”.This task will be performed using two methods, from now on - the Just and Regex methods.

4.1.1 From Questions to Queries - 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. Figure 4.1 depicts the process that accomplishes this task.

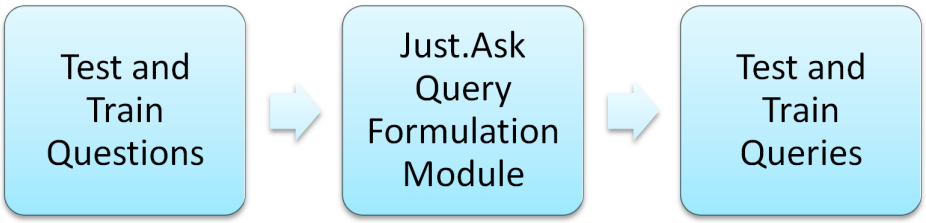

Figure 4.1: Just Method process to transform the datasets questions into queries.

The training and tests questions sets are used as input for the system and each question is processed trough the QA pipeline, as in a normal system run. The Passage Retrieval module utilizes the Keyword Query Formulator which uses a *keyword* oriented approach to transform each question into a query by removing question words, stop-words, function words and useless punctuation [1]. 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. For instance, the Just method produces the queries "old person order rent car Italy" and "apple water" for the questions "How old does a person have to be in order to rent a car in Italy ?" and "How much of an apple is water ?", respectively. These results support our belief that the use of a *keyword* oriented approach to convert questions into queries does not take into account how users formulate their queries, having one expected shortcoming - queries generated by this method may not represent accurately the queries that a user would do for some questions. To address this issue, a method that takes into consideration how users formulate their queries was devised - the Regex method.

The Regex method performs a matching between questions and a set of 100 regular expressions, that transform those questions into queries. The set of regular expressions was created based in empirical knowledge of how users usually make their queries in search engines. More specifically, for each fine category of the Li and Roth taxonomy, a set of regular expressions was created that corresponds to multiple question formulations of possible subjects under that category. The matching process is performed as follows: first, a set of regular expressions that possess a specific scope are matched with the question. If no match occurs, a set of general regular expressions are matched with the question. When a match

[1] Text between quotes remains unchanged due to its importance for the query target.

between question and a regular expression occurs, one or both of the following actions are performed: 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. Appendix A provides for each fine category of the Li and Roth taxonomy, an example of a question turned into query by this method.

Finally, it is important to mention that the set of regular expressions were developed in a generic manner so that they can be applied to other any questions datasets, being *dataset independent*. For the given training and test datasets, respectively, 95.89% and 94.4% of the questions are matched with regular expressions and converted into queries. The unmatched questions are usually questions that do not start with a question word such as:

- On what T.V. show could Tom Terrific be found ?
- CNN began broadcasting in what year ?
- Jude Law acted in which film ?
- The Kentucky Horse Park is close to which American city ?

Thus, in order to transform the unmatched questions in the datasets into queries, the Just approach is applied. It is not expectable that this decision will taint the overall results obtained by the Regex approach, since most of the questions in the datasets are matched with at least one regular expression.

4.1.2 Just Method Vs Regex Method - User Evaluation

As it was said before, the Regex method was developed as an improvement for the Just method, by taking into consideration how users formulate their queries and therefore producing queries that are more similar to the ones submitted by the users. To verify this assessment, three users evaluated the "quality" of the queries produced by both methods.

Each user was presented with a set of 50 questions and two sets of 50 queries each one corresponding to those same questions transformed into queries by the Just and Regex methods, respectively. The users evaluate each query using the following metrics:

- OK : The query possesses the necessary terms to precisely represent the information need that is embedded in the respective question. For instance, the query "percentage body muscle" have exactly the necessary terms to represent the question "What percentage of the body is muscle ?";
- M+ : The query possesses the necessary terms as well as unnecessary terms, i.e, terms that are redundant or that do not need be in the query in order for this to precisely represent the information need that is embedded in the respective question. The removal of these unnecessary terms

would contribute for a more precise representation of the information need that is embedded in the respective question. Using the same example, the query "percentage of the body is muscle" have unnecessary terms to represent the referred question such as "of", "the" and "is";

- M- : The query does not possess some terms that are necessary for the query to precisely represent the information need that is embedded in the respective question. Using the same example, the query "body muscle" does not possess the term "percentage" which is the headword of the referred question and, therefore, does not represent in a precise manner the information of that question.

It is important to mention that it is considered that these evaluation metrics form a ranking hierarchy, being OK the best evaluation a user can assign to a query, and being M- the worst evaluation a user can assign to a query. We consider M+ to be more important than M- since from a retrieval point of view, a query that does not have all the necessary terms may not retrieve as many relevant results as a query that possesses all the necessary terms (as well as unnecessary terms).

Just Method			
Measures	User 1	User 2	User 3
Queries classified as OK	37	26	37
Queries classified as M+	6	2	6
Queries classified as M-	7	22	7
Regex Method			
Measures	User 1	User 2	User 3
Queries classified as OK	14	39	36
Queries classified as M+	34	7	12
Queries classified as M-	2	4	2

Table 4.1: Just and Regex methods user evaluation results.

Table 4.1 depicts for each evaluation metric, the number of queries that the users assigned that evaluation metric. It can be seen from the total number of queries classified with OK, that the users consider the Just method as the method that produces the best queries (it possesses 100 queries classified with OK while the Regex method only possesses 89). When considering the queries classified with OK for each user individually, we can see that User3 gives a similar classification for both methods while User1 and User2 have almost antagonistic evaluations for both methods. Similarly, these two users have also antagonistic evaluations for the M- metric in the Just method, and the M+ metric for the Regex method.

A sample of a larger size would be necessary in order to achieve some realistic conclusions regarding how users elaborate their queries. However, taking into account the sample at hand, the data presented in the previous table seems to indicate the existence of three distinct types of users:

- Keyword users - Prefer *keyword* oriented queries - queries with less terms that are based on the most important content words of the question. These users prefer the queries generated by the Just method, since this method uses a keyword oriented approach to produce its queries. In this evaluation, User1 fits perfectly into this description;

- Near-Question users - Prefer more descriptive queries that have more terms (when compared to keyword queries) and for that reason, those queries slightly resemble (in most of the cases) syntactical well formed questions. These users prefer the queries generated by the Regex method, since this method produces queries that are a middle term between keyword queries and syntactical well formed questions. User2 is a perfect example of this kind of user;

- Mix users - Do not have a specific tendency to prefer keyword queries over more descriptive queries that have more terms, or vice versa. They recognized that for some questions a keyword query is a more suitable representation, while for others, a more descriptive query seems to be the most suitable representation. User3 can be seen as an example of this type of user, since it classified with OK almost the same number of queries for the Just and Regex methods (37 and 36 queries respectively);

User classification agreement		
Measures	Just Method	Regex Method
P_o\Fleiss Kappa for all metrics	44%\0.24	30%\0.13
P_o\Fleiss Kappa for OK queries	52%\0.3	34%\0.1
P_o\Fleiss Kappa for M+ queries	36%\0.06	80%\0.16
P_o\Fleiss Kappa for M- queries	90%\0.51	54%\0.2
P_o\Cohen Kappa for User1 and User2	60%\0.3	32%\0.14
P_o\Cohen Kappa for User2 and User3	54%\0.23	60%\0.33
P_o\Cohen Kappa for User1 and User3	68%\0.29	42%\0.13

Table 4.2: User classification agreement over the several metrics and between users.

To complement the information provided by Table 4.1, Table 4.2 depicts the agreement level of the users for all evaluation metrics and between the users themselves, using three measures: percentage of overall agreement, Cohen Kappa and Fleiss Kappa. The percentage of overall agreement (P_o) is the proportion of queries for which the users agree. The Cohen Kappa (Cohen, 1960) is a statistical measure of agreement between two raters when assigning categorical ratings to a number of items, while the Fleiss's Kappa (Fleiss et al., 1969) is an extension of Cohen's kappa to evaluate the level of agreement between more than two raters. Both Kappa measures are considered to be more robust than the percentage of overall agreement, since they take into account the agreement occurring by chance.

We can see that for queries classified as OK, the Just method has a Fleiss Kappa three times superior to the Fleiss Kappa of the Regex method, and also a superior P_o. Although both kappa values

indicate fair agreement by the users, the kappas and the P_o values confirm once again that users find the Just method as the method which produces more adequate queries. For the queries classified as M+, although the Fleiss Kappa value of the Regex method indicates only slight agreement by the users, is two times superior to the Fleiss Kappa value of the Just method that indicates poor agreement by the the users. This difference between the kappa values of the the two methods combined with the difference in their respective P_o's, allows us to conclude that users find the Regex method as the method that produces more queries with unnecessary terms. The scenario for the queries classified as M-, is the complete opposite of this last scenario. The Fleiss Kappa of the Just method indicates a moderate agreement by the users and it is two superior to the Fleiss Kappa value of Regex method that indicates only fair agreement by the users. Based on this difference and also in the difference of the respective P_o's of both methods, we can assess that users find the Just method as the method that produces more queries from whom are missing necessary terms, for representing the questions accurately.

Regarding the agreement level of the users between themselves, we can see for the Just method that there is a fair and similar inter-agreement between all users, since all Cohen Kappa values are similar between themselves and indicate a fair agreement, and also due to the slight similarity between the P_o values. As for the Regex method it can be seen from the Cohen Kappa and P_o values that the highest level of inter-agreement is obtained between User2 and User3, although this level of agreement is only fair. The remaining inter-agreements between users are similar and have only a slight level of agreement.

From the analysis of the user evaluation performed throughout this section three main conclusions can be made. First, users consider the Regex method as the method that produces more queries with unnecessary terms. Second, users find the Just method as the method that produces more queries from who are missing necessary terms, for representing the questions accurately. Finally the most important conclusion of this analysis, users consider Just method as the method that produces the most adequate queries.

4.2 Feature Engineering in Query Classification

The Query Classification task using the datasets obtained with the Regex and Just method, is performed with the same setting of the Question Classification evaluation described in Section 3.1.3. This is due to the fact that, as in Question Classification, modeling the Query Classification using SVM with a linear kernel and using the one-versus-all-multi-class-approach, achieves the best results. Experiments were conducted using the remaining available machine learning techniques, as well as using different parameterizations and kernel types for the SVM model, to attain the previous conclusion.

As features for the Query Classification task, some of the features used for Question Classification are utilized, as well as some features that were developed for this specific task.

4.2.1 Lexical Features

All of Just.Ask lexical features used in Question Classification - *n*-grams and binary *n*-grams - are utilized for performing Query Classification. The motivation for this decision is simple - these features achieved good results in Question Classification (either in their individual use or combined with other features) and it is empirically expected that their use in Query Classification will also produce good results.

Throughout this work, it has been empathized that one of the main characteristics of queries is that they possess significantly less terms than questions. Since this is a distinguished characteristic of queries, it should be taken into account for Query Classification. Three features were devised with this propose:

- 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.

The shape of the query's tokens (number of tokens that are lowercased, uppercased, etc.) can provide useful information that may aid the Query Classification task. For example, queries that have only one uppercased token such as "CNN" and "AIDS", probably belong to the category ABBREVIATION_EXPANSION. Two features were developed in order to capture this information:

- 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 of the previous cases;
- Binary Word Shape - Indicates if a certain query's tokens belongs to any of the previous "shape" categories.

4.2.2 Morpho-Syntactic Features

From the set of morpho-syntactic features that Just.Ask utilizes for Question Classification - POS tags and headword - only POS tags are used for Query Classification. The usage of POS tags as features for Question Classification in other similar works (Bomhoff et al., 2005; Kang & Kim, 2003) motivated this decision. It is important to mention that Just.Ask extracts POS tags with a parser that is trained for

questions and not queries, which may lead to an incorrect tagging process. Therefore, POS tags are not use in their full extent in the context of Query Classification.

As described in Section 3.1.3, in Question Classification the solely use of the headword feature achieved an accuracy of 92.6% and 82.8% accuracy for coarse- and fine-grained classification. These results, combined with the fact that queries have fewer terms than questions (which enhances the importance of the headword to the query), seem to indicate that this feature would be useful for Query Classification. However, Just.Ask is only capable of extracting the headword for questions and therefore the headword feature cannot be used in Query Classification. In order to circumvent this issue, two features that try to capture the headword "essence" were developed:

- 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. For instance, let us consider the question "What is Australia's national flower?", where the word *flower* is the question headword, and one possible query for the same question "Australia national flower". Using this feature on the previous query would result in *Australia* being recognized as the question headword instead of *flower*, which is not correct. The POS tagger is used to identify the first query token that is a noun;
- 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. For queries with more than four tokens this consideration is less likely to be true. To illustrate both of these cases let us consider the questions "What color is Mr. Spock 's blood ?" and "What was Queen Victoria 's title regarding India ?" as well as the respective queries for each question "color Mr. Spock blood" and "Queen Victoria title regarding India".

4.2.3 Semantic Features

All of Just.Ask semantic features used in Question Classification - named entities and semantic headword - are utilized for performing Query Classification. Regarding named entities, its use as feature for Query Classification presents some limitations since two of three approaches that Just.Ask uses to extract named entities were developed only considering questions. So likewise POS tags, named entities are also not use in their full extent in the context of Query Classification.

To avoid terminology misunderstandings, it is important to mention that in the context of Query Classification the semantic headword is going to be designated as "Pseudo Semantic headword". This is due to fact that as opposed to Question Classification where the actual question headword is enriched with semantics, in Query Classification the pseudo headword is enriched with semantics. As it was discussed before, the pseudo headword may not correspond to the actual headword of a query.

4.2.4 Approach to Feature Combination

Greedy forward feature selection is used in the defined setting to assess the individual and combined contributions of all the aforementioned features. Greedy forward feature selection consists in iteratively picking the feature that – in union with the features selected so far - maximizes a quality function q. In this context we consider q to be the accuracy attained by the set of selected features. The greedy forward feature selection algorithm works as follows: the first iteration analyzes the individual contribution of each feature and identifies the best K individual features. In the remaining iterations of the algorithm, the K sets of features that achieved the best accuracy in the previous iteration, are combined with all individual features. This process is repeated until the best accuracy that iteration N attains is equal or worse to the best accuracy attained by iteration N-1. For every iteration it is considered K = 3, since using a smaller value could cause the algorithm to discard possible good features for classification. Using these parameters the algorithm needs three iterations to perform the respective feature combinations for coarse-grained classification and two iterations for fine-grained classification.

4.2.5 Coarse-Grained Classification Results

The results of all the feature combinations for the Just and Regex methods, when considering coarse-grained classification, are detailed in Section B.1 of Appendix B. Table 4.3 presents, for coarse-grained classification, the most relevant results that ultimately lead to the best feature combination for both Just and Regex methods. The results are divided into three parts where each part represents the results obtained in each iteration of the feature combination algorithm. Table 4.3 also presents, for comparative purposes, the results obtained for Question Classification when using the same features.

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. Interestingly, these are also the best individual features for Question Classification from the set of features considered.

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.

Accuracy for coarse-grained classification			
Features	Regex Method	Just Method	Questions
First Token	48%	64%	45.6%
Binary Word Shape	35.6%	36.4%	18.8%
POS tags	39.4%	38.2%	59.6%
Ner-Replace	83.8%	74%	88.2%
Unigrams	83.8%	71.8%	88%
Binary Unigrams	84.2%	72%	88.2%
Bigrams	59.2%	33%	85.6%
First Token + Ner-Replace	85.6%	74.8%	88.6%
Binary Word Shape + Ner-Replace	83.8%	75.2%	88.6%
Unigrams + Binary Word Shape	83%	75.2%	88%
Unigrams + First Token	85.2%	74.6%	88%
Unigrams + POS tags	85%	72.8%	89.2%
Binary Unigrams + Binary Word Shape	82.8%	74.6%	88%
Binary Unigrams + First Token	85%	75%	87.6%
Binary Unigrams + POS tags	85%	72.4%	**89.6%**
Unigrams + Binary Word Shape + First Token	84.6%	**78.6%**	88%
Unigrams + POS tags + First Token	**86%**	74.6%	89.2%

Table 4.3: Accuracy of Just and Regex methods for coarse-grained classification.

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. An interesting fact is that none of the feature combinations that attain the best results for Just and Regex methods leads to the best results for Question Classification, which are obtained by the use of Binary Unigrams and POS tags features. The use of these features attains an accuracy of 89.6% which is inferior in 5.4% to the state of the art accuracy (95%) obtained with the use of Unigrams, Headword and Semantic Headword.

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.

4.2.6 Fine-Grained Classification Results

The results of all the feature combinations for the Just and Regex methods, when considering fine-grained classification, are detailed in Section B.2 of Appendix B. Similarly to the previous section, Table 4.4 presents for fine-grained classification the most relevant results that ultimately lead to the best feature combination for both Just and Regex methods, as well as the results obtained for Question Classifi-

cation when using the same features.

Accuracy for fine-grained classification			
Features	Regex Method	Just Method	Questions
First Token	34.6%	36.4%	46.6%
Ner-Replace	78.8%	68.6%	82.4%
Binary Word Shape	33.2%	33.6%	10.4%
Number of Tokens	32%	30.8%	31.4%
Unigrams	78.4%	67.6%	80.2%
Binary Unigrams	78.4%	67.6%	80.4%
Bigrams	50%	23%	73.8%
Binary Word Shape + Ner-Replace	**80.2%**	**69.6%**	**82.8%**
Unigrams + Ner-Replace	79%	66.4%	83%
Unigrams + Binary Word Shape	78.6%	69.2%	80.6%
Unigrams + Number of Tokens	78.2%	68.2%	80.2%

Table 4.4: 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%. These are also the best individual features for Question Classification from the set of considered features.

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. It is important to note, that once again, none of the feature combinations that attain the best results for Just and Regex methods, leads to the best results for Question Classification, which are obtained by the use of Binary Unigrams and Ner-Replace features. The use of these features attains an accuracy of 82.8% which is inferior in 7.6% to the state of the art results (90.4%) obtained with the use of Binary Unigrams, Headword and Semantic Headword.

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.

4.3 Building Ensemble Classifiers for Query Classification

The previous section describes the contribution of several features in training a SVM classifier (with a linear kernel) to classify queries. Having the winning solution of the KDD-Cup as main motivation, several ensemble classifiers were built with different strategies, in order to improve the results described in the previous section for both Just and Regex methods.

An ensemble is a set of models whose predictions are combined by weighted averaging or voting. Dietterich states that "A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse." (Dietterich, 2000). Taking this statement into consideration, the ensemble classifiers are built upon three types of classifiers: two SVM classifiers with linear and Radial Basis Function (RBF) kernels respectively, and a Naive Bayes classifier. Considering a SVM classifier that uses a linear kernel to be a different classifier from a SVM classifier with a RBF kernel is clearly an abuse of notation, since they both share the same classifier type and only differ in one parameter. However, we consider them do be distinct classifiers due to fact that the use of different kernels results in different performances, like it would probably happen if two "true" distinct classifiers were used.

4.3.1 Approach to Classifier Combination

For each type of classifiers previously mentioned - SVM classifiers with linear and RBF kernels respectively and a Naive Bayes classifier - we consider its six best feature combinations[2], that we denote as *base classifiers*. In order to identify the base classifiers for each type of classifiers, each one of these was trained and tested using two pairs of datasets of queries obtained by the Just and Regex methods, respectively. More specifically, for each method, each base classifier is trained with a dataset of 5000 queries using the approach to feature combination described in Section 4.2.4, and tested with dataset of 500 queries. The 500 queries used for testing correspond to 500 random queries of the training dataset with 5500 queries used in the evaluation of the SVM query classifier (Section 4.2) that we want to improve. The training dataset for each base classifier is composed by the remaining 5000 queries from the SVM query classifier training dataset. Appendix C depicts, for the Just and Regex methods, the six base classifiers for each type of classifier for both coarse and fine grained classification. The base classifiers are combined into an ensemble classifier by using one of the following voting methods:

- Simple Agreement - The category which has more votes from each base classifier is assumed as the correct one. Hence, this voting method considers that the votes from all base classifiers have all

[2]The feature combinations that lead to the six best accuracy results.

the same weight. This weighting process may not be always valid since different base classifiers have different performances;

- Total Accuracy Agreement - As before, the category which has more votes from each base classifier is assumed as the correct one. The vote of each base classifier has a weight proportional to the total accuracy of that same base classifier for the test dataset. Therefore, this method is an improvement over the previous method, since it takes into the consideration, in the weighting process, the different performances of the base classifiers. However, a base classifier may achieve a high accuracy for certain categories while having low accuracy for others. This indicates that assigning a weight to a vote of a base classifier solely based on its global accuracy is not always valid either, which led us to the next voting method;

- Category Accuracy Agreement - Likewise, the category which has more votes from each base classifier is assumed as the correct one. The vote of each base classifier has a weight proportional to the total accuracy of that same base classifier obtains for a certain category. For instance, if for the test dataset a base classifier attains a accuracy of 90% for the DESCRIPTION_DEFINITION category, then that base classifier will have a weight of 0.9 for each query that it classifies with that category. Thus, this method solves the main disadvantages of the previous two voting methods. The successful use of this method in the ensemble classifiers of the winning solution of the KDD-Cup, served as the main motivation for devising this method.

By utilizing one of the previous voting methods, an ensemble classifier can be obtained at three distinct levels: individual, pair and group levels. At individual level, only the base classifiers (the six best feature combinations) of a type of classifiers are combined into an ensemble classifier. The pair level, as the name suggests, combines the base classifiers belonging to a pair of type of classifiers. Finally, the group level combines all the base classifiers belonging to all the three types of classifiers considered.

4.3.2 Experimental Setup

For each voting method presented in the previous section, a ensemble classifier was obtained at individual, pair and group levels. In order for a comparison between the SVM query classifier evaluated earlier and the ensemble classifiers to be made, the latter were tested using the same two datasets of 500 queries obtained, respectively, by the Just and Regex method respectively, that were utilized in the SVM query classifier evaluation. The result comparison between the SVM query classifier and the ensemble classifiers, only possible by using the same training and testing data, was the main driver for using different testing data (from the datasets mentioned in this section) to test the base classifiers. If the same testing data were to be utilized for both the base classifiers and ensemble classifiers, there could be some biasing of the results attained by the ensemble classifiers. As for the training data, we have already depicted

that the training dataset of the base classifiers is composed by 5000 queries from the 5500 queries used in the SVM query classifier training dataset. Although the training data used to train the base classifiers and the SVM query classifier is not exactly the same, the results attained by the ensemble classifiers can still be compared with the results of the SVM query classifier.

4.3.3 Coarse-Grained Classification Results

Table 4.5 presents, for coarse-grained classification, the accuracy of the ensemble classifiers obtained at individual, pair and groups levels, for each of the voting methods considered.

Accuracy for coarse-grained classification			
Voting Method	Ensemble Classifiers	Regex Method	Just Method
N.a	**SVM Query Classifier Baseline**	**86%**	**78.6%**
Simple Agreement	**SVM-Linear**	85.6%	**77.8%**
	SVM-RBF	75%	64.4%
	Naive Bayes	71.2%	67.6%
	SVM-Linear + SVM-RBF	**86.2%**	74.2%
	SVM-Linear + Naive Bayes	85.4%	75.2%
	SVM-RBF + Naive Bayes	77.2%	69.6%
	SVM-Linear + SVM-RBF + Naive Bayes	85.4%	75.2%
Total Accuracy Agreement	**SVM-Linear**	85.8%	**77.8%**
	SVM-RBF	75%	64.4%
	Naive Bayes	72.2%	67%
	SVM-Linear + SVM-RBF	**86.2%**	74.2%
	SVM-Linear + Naive Bayes	85.4%	75.2%
	SVM-RBF + Naive Bayes	75.4%	70%
	SVM-Linear + SVM-RBF + Naive Bayes	85.4%	75.4%
Category Accuracy Agreement	SVM-Linear	85%	76.6%
	SVM-RBF	73.8%	64.6%
	Naive Bayes	71.2%	67%
	SVM-Linear + SVM-RBF	85.4%	72.6%
	SVM-Linear + Naive Bayes	81.2%	71.4%
	SVM-RBF + Naive Bayes	75.6%	68.6%
	SVM-Linear + SVM-RBF + Naive Bayes	81.8%	74%

Table 4.5: Accuracy of all ensemble classifiers for coarse-grained classification.

It can be seen that all ensemble classifiers achieve, as expected, an higher accuracy than the individual accuracy of its composing base classifiers (depicted in Sections C.1 and C.2 of Appendix C). However, the results attained by the ensemble classifiers fell short of expectations since in a general manner they marginally improve, or do not improve at all, the results obtained by the previously evaluated SVM query classifier. This classifier possesses a maximum accuracy of 78.6% for the Just method, while the "SVM-Linear" ensemble classifiers generated by the Simple and Total Accuracy Agreement voting methods, achieve their best results for the Just method with an accuracy of 77.8%. For the Regex method, 86.2% is the maximum accuracy obtained by the ensemble classifiers "SVM-Linear + SVM-RBF" generated by the Simple and Total Accuracy Agreement voting methods. This result represents

an improvement of 0.2% over the maximum accuracy of 86% attained by the SVM query classifier for the Just method. Since this cannot be considered a significant improvement, we still consider the SVM query classifier to be the best query classifier for coarse-grained classification using both Just and Regex approaches.

Another unexpected fact, is that the best results for both Just and Regex methods are obtained through the Simple and Total Accuracy Agreement methods and not trough the Category Accuracy Method as it was expected. As it was mentioned previously, the Category Accuracy Method solves the main disadvantages of Simple and Total Accuracy Agreement methods voting methods, so it was expectable that its use led to better results when comparing to the previous two methods.

4.3.4 Fine-Grained Classification Results

Table 4.6 presents for coarse-grained classification the accuracy of the ensemble classifiers obtained at individual, pair and groups levels, for each of the voting methods considered.

Accuracy for fine-grained classification			
Voting Method	Ensemble Classifiers	Regex Method	Just Method
N.a	**Evaluated SVM Query Classifier**	**80.2%**	**69.6%**
Simple Agreement	SVM-Linear	77%	67%
	SVM-RBF	59.2%	47%
	Naive Bayes	58.2%	48.8%
	SVM-Linear + SVM-RBF	74.8	64.8%
	SVM-Linear + Naive Bayes	**77.8%**	68.6%
	SVM-RBF + Naive Bayes	62%	50.6%
	SVM-Linear + SVM-RBF + Naive Bayes	74%	60.6%
Total Accuracy Agreement	**SVM-Linear**	77%	**68.8%**
	SVM-RBF	59.4%	46.4%
	Naive Bayes	58%	48.6%
	SVM-Linear + SVM-RBF	75%	66.4%
	SVM-Linear + Naive Bayes	**77.8%**	68.2%
	SVM-RBF + Naive Bayes	61.4%	51.8%
	SVM-Linear + SVM-RBF + Naive Bayes	74.4%	63.4%
Category Accuracy Agreement	SVM-Linear	77.4%	67%
	SVM-RBF	59.4%	47%
	Naive Bayes	57%	48.8%
	SVM-Linear + SVM-RBF	74.2%	64.8%
	SVM-Linear + Naive Bayes	74%	68.6%
	SVM-RBF + Naive Bayes	61.6%	50.6%
	SVM-Linear + SVM-RBF + Naive Bayes	67.4%	60.6%

Table 4.6: Accuracy of all ensemble classifiers for fine-grained classification.

As in coarse-grained classification, the results attained by the ensemble classifiers fell short of expectations since they do not improve at all the results obtained by the previously evaluated SVM query classifier. This classifier possesses a maximum accuracy of 69.6% for the Just method, while the "SVM-

Linear" ensemble classifier generated by the Total Accuracy Agreement voting method, achieve their best results for the Just method with an accuracy of 68.8%. For the Regex method, 77.8% is the maximum accuracy obtained by the ensemble classifiers "SVM-Linear + Naive Bayes" generated by the Simple and Total Accuracy Agreement voting methods. This result does not improve the maximum accuracy of 80.2% attained by the SVM classifier for the Regex method. Also analogous to the coarse-grained classification, the best results for both Just and Regex methods are obtained through the Simple and Total Accuracy Agreement methods and not trough the Category Accuracy Method as it was expected.

Despite an improvement of 0.2% for the Regex method in coarse-grained classification, all the results attained for both coarse and fine-grained classification by all ensemble classifiers are always inferior to the results of the SVM query classifier. Since the mentioned improvement is not significant and can be neglected, we can state that the SVM query classifier still remains as the best query classifier for both coarse and fine-grained classification with both Just and Regex methods.

4.4 Simultaneous Query and Question Classification

All the work described throughout this chapter 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.

4.4.1 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 using the algorithm for feature combination described in Section 4.2.4. The queries in the train dataset correspond to the questions of the train and test datasets used in the evaluation of the SVM question classifier (Section 3.1.3) converted into queries by the Just method. The questions in the train dataset used in the evaluation of the SVM question classifier, are also the same questions used in the train dataset of the Query and Question Classifier. It is important to mention that the Just method was utilized to convert questions into queries, since users consider the queries generated by this method represent more accurately the queries that they would do to Just.Ask (as seen in Section 4.1.2).

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. A question can be formulated using very different formulations, but most of these formulations often begin with a question word. Since queries usually do not begin with a question word, the question word in the beginning of the user input can be used to distinguish questions from queries. On the other hand, as has been empathized throughout this work, queries often possess less terms that questions, hence the number of terms of the user input can also be used to distinguish questions from queries.

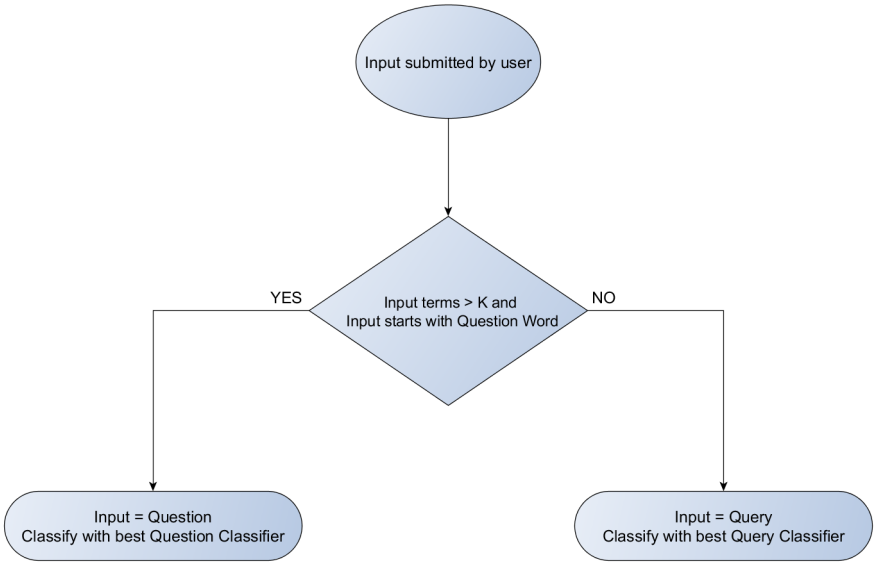

Figure 4.2: Procedure to distinguish the two types of user input.

These two characteristics are used in a simple procedure depicted in Figure 4.2: if the user input starts with a question word and it has more than K terms[3], 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. Since the two aforementioned characteristics cannot be used in a deterministic manner to distinguish questions from queries, some of the user input may be misclassified, i.e, questions may be considered a query and vice versa.

[3]We consider K to be the number of terms of the lengthiest query in the training dataset used in the first approach.

4.4.2 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 the SVM question classifier and to these same questions converted into queries by the Just method. Additionally, both the best query and question classifiers 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. It must be kept in mind that both the best query and question classifiers have different parameterizations for coarse and fine-grained classification, namely:

- Best Query Classifier for Coarse Grained Classification - Unigrams, Binary Word Shape and First Token are used as features;
- Best Question Classifier for Coarse Grained Classification - Utilizes Unigrams, Headword and Semantic Headword as features;
- Best Query Classifier for Fine Grained Classification - Binary Word Shape and Ner-Replace are used as features;
- Best Question Classifier for FIne Grained Classification - Utilizes Binary Unigrams, Headword and Semantic Headword as features.

Table 4.7 presents for coarse and fine-grained classification the accuracy of all considered classifiers.

Accuracy for coarse and fine-grained classification		
Classifiers	Coarse-Grained	Fine-Grained
Best Query Classifier	81.9%	74.7%
Best Question Classifier	80.3%	73.4%
Query and Question Classifier	82.7%	76.3%
Most Adequate Query and Question Classifier	**86.6%**	**79.7%**

Table 4.7: Accuracy of simultaneous query and question for coarse and fine-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.

Another interesting fact is that the Best Query Classifier slightly outperforms the Best Question Classifier, which points out that the performance of query classifier in classifying questions is slightly better that the performance of question classifier in classifying queries. One possible explanation for this occurrence is that the Headword and Semantic Headword features used by the Best Question Classifier in both classification types are extracted using manual patterns specifically devised for questions which make these features not useful in query classification (as shown in Sections 4.2.5 and 4.2.6).

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.

5 Query Reformulation and Query Expansion

This chapter is devoted to the creation of Query Reformulation and Query Expansion mechanisms in Just.Ask system, so that more relevant passages can be retrieved by the system. The chapter is divided into four sections that can be perceived as three coarse parts.

The first part, Section 5.1 compares Just.Ask's baseline retrieval results with the results obtained when the Regex Method, described in the previous chapter, is used as query formulator.

Sections 5.2 and 5.3 constitute the second part. These sections describe two query formulators - the Query Reformulation and Query Expansion Formulators - that implement Query Reformulation and Query Expansion mechanisms, respectively, as well as the results obtained when utilizing each one of those query formulators.

Finally, the last part which corresponds to Section 5.4, depicts the results obtained by Just.Ask when the query formulations mentioned in the last paragraph are used simultaneously.

5.1 Regex Query Formulator - A New Query Formulator

Although the Regex method was devised for the Query Classification task as stated in Section 4.1.1 from Chapter 4 , it can clearly be used as a query formulator for Just.Ask's Passage Retrieval Module. In order to maintain the terminology used for the query formulators consistent, the Regex method will be denoted as "Regex Query Formulator" throughout this chapter. Using the evaluation setting and evaluation metrics defined in Section 3.2.1 (183 factoid questions and two different corpora for answers accessed by two search engines, respectively), the Passage Retrieval module was tested using the Regex Query Formulator as query formulator. Table 5.1 presents the results of this evaluation as well as the Passage Retrieval module baseline results (obtained using the Keyword Query Formulator) for comparative purposes.

It can be seen that the results attained by the two query formulators are similar in an overall manner. When considering Bing search engine, the use of the Regex Query Formulator leads to a decrease of 0.03 and 0.05 over the baseline values of MRR_{ALLQ} and $\text{MRR}_{QPOSPASSAGES}$, respectively. On the other hand when considering Lucene search engine, the results obtained with the Regex Query Formulator are identical to the baseline results except in the number of questions for each there is at least one positive

Passage Retrieval Baseline - Keyword Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	137 (74.9%)	0.37	0.5
Lucene	102 (55.7%)	0.19	0.33
Regex Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	137 (74.9%)	0.34	0.45
Lucene	103 (56.3%)	0.19	0.33

Table 5.1: Passage Retrieval baseline and Regex Query Formulator results.

passage (#Question$_{1+PosPassage}$), where one more question with at least one positive passage is obtained over the baseline value. From this analysis we can conclude that the Keyword Query Formulator still remains as the best query formulator to use with Bing search engine. We can also conclude that both query formulators are suitable to use with Lucene search engine, since the slight improvement that Regex Query Formulator obtains is not significant and can be neglected.

Regardless of the previous conclusions, both query formulators were utilized for both Bing and Lucene search engines in the experiments described in Sections 5.2 and 5.3. The motivation for this decision was that the inclusion of these formulators in more complex query formulation techniques, such as query reformulation and query expansion, could have a different impact on the performance of those techniques, and therefore should not be restricted.

5.2 Query Reformulation Formulator

As previously stated, Query Reformulation is a technique that can be used to improve the quality of a query in a QA system, by removing/adding/reweighting query terms. As depicted in Sections 2.2.4.1 and 2.2.4.4, Open-Ephyra and Aranea QA systems use a Query Reformulation technique which consists in using queries that are possible reformulations of the user question and that try to anticipate the location of the answer in order to extract it. The motivation for this technique is to take 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.

A previous work by the author (Pires, 2012) performed an extrinsic and intrinsic evaluation of Open-Ephyra and Just.Ask systems under the same evaluation setting, and reached some relevant conclusions regarding the performance of the Passage Retrieval module of the two systems. The most important conclusion in this regard, is that the performance of Open-Ephyra's Passage Retrieval module surpasses the performance of Just Ask's module due to use of the aforementioned Query Reformulation technique. These results attained by Open-Ephyra were the main motivation to endow Just.Ask with the

same Query Reformulation mechanisms. The fact that Open-Ephyra is an open source system also motivated this decision, since by examining the source code used to implement the Query Reformulation technique one can more easily implement the same technique in a different system.

5.2.1 Just Ask Query Reformulation Approach

Before describing how the query reformulation technique was implemented in Just.Ask, it is import to describe how Open-Ephyra implements that technique since it is one of the main components of Just Ask's query reformulation implementation. Query Reformulation is accomplished in Open-Ephyra trough the utilization of 13 manually devised patterns that take into account the most common formulations that questions have (these patterns are depicted in Section D.1 of Appendix D). Each time a pattern matches a question at the lexical level, one or more queries with reformulations of the question are generated. For instance, for the question "Who is the Japanese Prime Minister ?" Open-Ephyra generates two query reformulations: ""is the Japanese Prime Minister"" and ""the Japanese Prime Minister is"". The extra pair of quotation marks in each of the previous two reformulations represents the phrase operator in most of the existing search engines and it is used to ensure that search engines only retrieve passages that contain, exactly, the phrase inside the extra quotation marks.

Just Ask takes the query reformulation approach of Open-Ephyra one step further, by incorporating that approach in a more complex procedure with the same purpose. The "Query Reformulation Formulator" executes this procedure that is composed by three steps: the Category Level Matching, High Level Matching and Alternative Formulation steps.

5.2.1.1 Category Level Matching Step

The submitted user question is matched with a set of patterns that are specific to the category that was assigned to that question by the Question Classifier in the Question Interpretation module. By having a set of patterns that were manually devised for each fine category of the Li and Roth taxonomy (resulting in a total of 150 patterns for all the categories), we expect to take advantage of some vocabulary that is specific to a certain category, by using that vocabulary to generate more queries with reformulations of the question. To corroborate this point, let us consider the question "How long did the Charles Manson murder trial last ?" that belongs to the category NUMERIC_PERIOD. Since it is known that the answer of the question is a numeric period, we can use several question reformulations of the form "the Charles Manson murder trial lasted X Y" where X is a number in its written or numeric form, and Y is a time period such as hours, days, weeks, months, years, etc.

When a match between a question and a pattern at the lexical level occurs, at least one pair of queries with reformulations of the question is generated. Each pair of generated queries is composed

of a *phrase* and a *standard* query. Phrase queries are identical to the queries with reformulations that Open-Ephyra produces, i.e, each query reformulation is put inside quotation marks (phrase operator) to ensure that search engines only retrieve passages that contain, exactly, the phrase inside the quotation marks. Standard queries are identical to phrase queries except that they do not have quotation marks surrounding the contents of the query. Using standard queries, search engines retrieve passages that contain all the query terms without a specific order between the terms, in opposition of what happens in phrase queries. The motivation for the use of both of these query types lies in a simple fact - the answer to a question can be formulated in multiple ways and it is not possible to know all the possible formulations of the answer to a question. Thus, in the worst case scenario, phrase queries cannot retrieve any passages at all or retrieve passages that are not relevant for the question at hand. In this scenario standard queries are used as a backup plan since they augment the search scope by being less restrictive than phrase queries, which results in an increase of the number and quality of the passages retrieved.

Last but not least, it is important to mention that, although performing a matching between questions and patterns at the fine category level can have some advantages as mentioned in the first paragraph of this section, it would be useless to do so if the category assigned to question by Just.Ask's Question Classifier was the wrong one. However, this is not a concern in this case since Just.Ask's Question Classifier has a accuracy of 90.4% for fine classification, which represents a reasonable value for misclassified questions.

Section D.2 of Appendix D depicts some of the patterns devised for this step as well as the queries generated by those patterns.

5.2.1.2 High Level Matching Step

This step is reached when no matches have occurred between a question and the category oriented patterns used in the previous section. In this step we match the submitted user question with Open-Ephyra's 13 manually devised patterns that take into account the most common formulations that questions have. Like in the Category Level Matching Step, when a match between a question and a pattern at the lexical level occurs, at least one pair of queries with reformulations of the question is generated, being each pair of queries composed of a phrase and standard query. For example, using these patterns the question "What is the purpose of a car bra ?" would generate the following phrase and standard queries, respectively: ""the purpose of a car bra is"" and "the purpose of a car bra is".

5.2.1.3 Alternative Formulation Step

In a similar fashion as before, this step is reached when no matches have occurred between a question and the patterns used in the Category Level Matching and High Level Matching steps. In this step the

submitted user question is transformed into a query by using one of the remaining query formulators of Just.Ask - the Keyword Query Formulator or the Regex Query Formulator.

5.2.2 Experimental Setup

Using the evaluation setting and evaluation metrics defined in Section 3.2.1, the Passage Retrieval module was tested using the Query Reformulation Formulator as query formulator in two different tests: the matched questions and the overall tests.

The matched questions test, as the name suggests, compares the performance of the Query Reformulation Formulator with the Keyword and Regex Query Formulators, for the questions of the test dataset that matched a regular expression of the Query Reformulation Formulator. Since the Keyword and Regex Query Formulators are used by the Query Reformulation Formulator when no matching between a question and regular expressions occurs, this test was necessary in order to allow a comparison of the results obtained by all formulators in a non biased manner, which would not be possible if all the questions of the test dataset were to be considered.

The overall test is similar to the previous one, being the only difference the fact that the comparison between the formulators is done with all the questions in the test dataset. This test allows us to analyze the contributions of the Query Reformulation Formulator for the retrieval task in an overall manner and to compare these results with the overall results attained by the Keyword and Regex Query Formulators.

5.2.3 Query Reformulation Matched Questions Results

Table 5.2 presents, for all the query formulators, the results for the 143 questions (78.1%) of the 183 questions of the test dataset that matched a regular expression of the Query Reformulation Formulator. It must be kept in mind that we use 50 passages as the maximum number of passages to retrieve per query, which means that since the Query Reformulation Formulator produces at least two queries for each question, this method ends up frequently retrieving more that 50 passages in total (in opposition to the remaining methods).

It can be seen that the Query Reformulation Formulator results surpasses the other formulators results in both search engines and in all evaluation metrics. When considering Bing search engine, the use of the Query Reformulation Formulator leads to an increase of five questions (3.5%) in the #Question$_{1+PosPassage}$ and an increase of 0.1 and 0.12, respectively, over the largest value of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ attained by the other formulators. For Lucene search engine the improvements of 0.05 and 0.06 over the largest value of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ attained by the other formulators, respectively, represent approximately half of the improvements achieved with Bing

Passage Retrieval Baseline - Keyword Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	108 (75.5%)	0.37	0.48
Lucene	78 (54.5%)	0.17	0.31
Regex Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	108 (75.5%)	0.32	0.43
Lucene	78 (54.5%)	0.18	0.34
Query Reformulation Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	113 (79%)	0.47	0.6
Lucene	84 (58.7%)	0.23	0.39

Table 5.2: Keyword, Regex and Query Reformulation Formulators results for the 143 matched questions.

search engine for the same evaluation metrics. On the other hand, the increase of six questions (4.2%) in the #Question$_{1+PosPassage}$ is very similar to the improvement presented with Bing search engine. Nonetheless, all these results corroborate once again that the query reformulation approach works better for information sources with large data redundancy (the Web) than information sources with less data redundancy (paragraphs from the NYT from 1987 to 2007).

Based on the presented results we can conclude that for questions that match regular expressions from the Query Reformulation Formulator, this same formulator outperforms the Keyword and Regex Query Formulators, thus being the best query formulator of Just.Ask for those questions (and consecutively for all questions as we will demonstrate in the next section).

5.2.4 Query Reformulation Overall Results

In order to analyze the global contribution of the Query Reformulation Formulator, Table 5.3 presents for each query formulator its performance when all the questions in the test dataset are considered. Thus, the Query Reformulation Formulator used the Keyword and Regex Query Formulators to transform into queries, questions that did not match the formulator regular expressions.

As in the previous test, the Query Reformulation Formulator overall results surpasses the other formulators overall results in both search engines and in all evaluation metrics. This fact was to be expected, since we have already depicted the superior performance of the Query Reformulation Formulator over the remaining formulators for questions that match regular expressions from that formulator, which represent 78.1% of the testing data.

When considering Bing search engine, the use of the Query Reformulation Formulator leads to an

Passage Retrieval Baseline - Keyword Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	137 (74.9%)	0.37	0.5
Lucene	102 (55.7%)	0.19	0.33
Regex Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	137 (74.9%)	0.34	0.45
Lucene	103 (56.3%)	0.19	0.33
Query Reformulation Formulator with Keyword Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	143 (78.1%)	0.45	0.57
Lucene	108 (59%)	0.23	0.39
Query Reformulation Formulator with Regex Query Formulator - 50 passages			
Search Engine	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing	143 (78.1%)	0.44	0.57
Lucene	109 (60%)	0.22	0.37

Table 5.3: Keyword, Regex and Query Reformulation Formulators overall results.

increase of six questions (3.2%) in the #Question$_{1+PosPassage}$ and an increase of 0.08[1] and 0.07[1], respectively, over the largest value of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ attained by the other formulators. Still in this regard, it can be noticed that use of the Keyword and Regex Query Formulators as "backup" formulators for the Query Reformulation Formulator produces almost identical results. The use of the Keyword Query Formulator leads to a improvement of 0.01 over the value of MRR$_{ALLQ}$ obtained when using the Regex Query Formulator, however, this slight improvement is not significant and can be neglected. Thus, when using the Bing search both Keyword and Regex Query Formulators are suitable backup formulators for the Query Reformulation Formulator.

For Lucene search engine, the use of the Query Reformulation Formulator leads to an increase of six questions (3.7%)[1] over the largest value of #Question$_{1+PosPassage}$ attained by the other formulators, and an increase of 0.04[1] and 0.06[1] , respectively, over the values of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$. As opposed of what was verified for Bing search engine, the use of the Keyword and Regex Query Formulators as backup formulators for the Query Reformulation Formulator produces slight different results for all evaluation metrics. Specifically, the Regex Query Formulator obtains one more more question with at least one positive passage than the Keyword Query Formulator and this last formulator possesses a improvement of 0.01 and 0.02 over the values of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ obtained by the Regex Query Formulator. We consider the difference in the #Question$_{1+PosPassage}$ to be significant since it results, in the best case scenario, in one more correct question answered when using the Regex Query Formulator, on the other hand, we also consider the difference between the MRR$_{QPOSPASSAGES}$ values of both formulators to have some relevance either. Due to these reasons, we consider that when using

[1]This value is obtained considering the biggest improvement obtained by the Query Reformulation Formulator.

the Lucene search both Keyword and Regex Query Formulators are suitable backup formulators for the Query Reformulation Formulator.

From this analysis we can conclude that the Query Reformulation Formulator is Just.Ask's best query formulation strategy until this moment, and that both the Keyword and Regex Query Formulators as backup for the Query Reformulation Formulator for both Bing and Lucene search engines.

5.3 Query Expansion Formulator - Expansion Using Wordnet

As previously stated, Query Expansion is a technique that can be used to improve the quality of a query in a QA system, by adding new query terms. From all the Query Expansion types and techniques described in Sections 2.2.2 and 2.2.3, we focused in AQE trough knowledge structures, where expansion terms are collected from previously developed knowledge structures that contain information about a set of terms (synonyms, related terms, etc.).

Specifically, our main focus is the general-purpose thesaurus known as Wordnet (Fellbaum, 1998). WordNet is a directed acyclic graph composed of word forms and meanings related to each other with a many to many relationship. As a general-purpose thesaurus Wordnet describes lemmas of single/multiple words grouped in synsets (sets of synonyms identifying concepts/senses) and that are interlinked by lexical/morphological and semantic relations. Synsets also contain other linguistic information that can be typically found on dictionaries such POS and common usage examples. Several studies (Voorhees, 1994; Hsu et al., 2008; Zhang et al., 2009) that show that Wordnet can be used to expand, and consecutively improve queries, served as main motivation for choosing Wordnet as knowledge structure.

5.3.1 Query Expansion Approach

Just.Ask uses the Java WordNet Library (JWNL) extension, JWNL*Simple* (Fialho et al., 2011), developed by L2F to leverage the Wordnet resources. Most of Java Wordnet Libraries provide methods for exploring semantic relationships that have to be conjugated in a complex manner in order for the developer to obtain the desired information. The main motivation to use the JWNL*Simple* extension is that, as its name suggests, JWNL*Simple* extends the existing retrieval procedures in the common Java WordNet Libraries by providing "straight forward" retrieval procedures that allow the developer to obtain the desired information in a simpler manner. The following JWNL*Simple* procedures were used to expand query terms with semantically related terms:

- 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”.

Specifically, these procedures are applied to the question headword/compound headword and to the main verbs that exist in the query at hand. The motivation for expanding the question headword/compound relies on the fact that the headword/compound headword represents the information that is being sought after, so by expanding it with semantic related terms, more relevant passages should be retrieved. As for expanding the main verbs of the query, the improvements observed in the Raposa system (Sarmento, 2008) due to the use of this technique (although not using Wordnet as knowledge structure) served as main motivation for applying it in Just.Ask.

The semantic related terms obtained trough the Direct Hypernym and Similar To procedures are then joined to the question headword/main verbs by an OR operator, producing a query in the Disjunctive Normal Format. To illustrate this process, let us consider the question “What do the “chasing arrows” symbolize ?” where the word *symbolize* is the question headword one possible headword expansion query could be ““chasing arrows” (symbolize OR intend OR mean OR represent)”. In this regard is also important to state that verb expansions are always retrieved in the infinitive form and therefore are converted to the appropriate form. For instance, if the verbs *grant* and *present* were considered as expansions for the verb *awarded* in the query “prize Elizabeth Neufeld awarded”, the expanded query would be “prize Elizabeth Neufeld (awarded OR granted OR presented)”.

Finally, it is important to mention that the “Query Expansion Formulator” that implements this approach, utilizes both the Keyword and Regex Formulators to generate the queries that will be expanded by Query Expansion Formulator.

5.3.2 Experimental Setup

Using the evaluation setting and evaluation metrics defined in Section 3.2.1, the Passage Retrieval module was tested using the Query Expansion Formulator. In order to evaluate the individual and conjugated contributions of expanding the question headword/compound headword and the main verbs of a query, three tests were made using different parameterizations for the number of expanded terms to be added to the query. These different parameterizations attempt to shed some light on how the number of selected expansion terms obtained through Wordnet impacts the retrieval effectiveness.

A conducted preliminary experiment showed that there is a large discrepancy in the number of expansion terms of different queries, with queries having from five expansion terms to queries having twenty five expansion terms. Due to this fact and in order to measure, in a weighted manner, the impact

of different number of expansion terms in the retrieval process, we chose not to use the standard parameterization method which consists of defining a threshold value for the number of expansion terms that can be added to all queries. With that purpose we have defined the number of expansion terms that can be added to a query using threshold values based on percentages that take into consideration the number of expansion terms that each query possesses. Specifically, we considered five threshold values - 20%, 40%, 60%, 80%, 100% - being that if the considered threshold value is 20% in a query that possesses 10 expansions terms, only two expansion terms would be added to that query.

5.3.3 Headword/Compound Headword Expansion Bing Results

Table 5.4 presents the Headword/Compound Headword expansion best results when Bing is used as search engine, as well as the results obtained by previously evaluated formulators for comparative purposes. All the results of the Headword/Compound Headword expansion when using Bing as search engine are detailed in Section E.1 of Appendix E.

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator	137 (74.9%)	0.37	0.5
Regex Query Formulator	137 (74.9%)	0.34	0.45
Query Reformulation Formulator with Keyword Query Formulator	143 (78.1%)	0.45	0.57
Query Reformulation Formulator with Regex Query Formulator	143 (78.1%)	0.44	0.57
Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
60%	**140 (76.5%)**	**0.40**	**0.53**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
80%	**129 (70.4%)**	**0.35**	**0.49**
100%	**129 (70.4%)**	**0.35**	**0.49**

Table 5.4: Headword/Compound Headword best expansion results when using Bing search engine.

The first consideration that comes to mind after analyzing the table is that the Headword/Compound Headword best expansion results do not improve at all the best results obtained so far using the Query Reformulation Formulator with both the Keyword and Regex Query Formulators. When considering the improvements over the Keyword Query Formulator we can see in fact improvements, since there is an increase of three questions (1.6%) in the #Question$_{1+PosPassage}$ and an increase of 0.03 over the values of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$. On the other hand, although a significant improvement of 0.05 over the MRR$_{QPOSPASSAGES}$ value obtained by the Regex Query Formulator exists, it can also be seen a significant decrease of eight questions (4.5%) in the #Question$_{1+PosPassage}$. Thus, we can conclude that expanding the Headword/Compound Headword for the queries generated

by the Regex Query Formulator improves the ranking of the retrieved relevant passages although being hazardous to the retrieval of more relevant passages.

Another interesting fact is that the Headword/Compound Headword best expansion results when using both Keyword and Regex Query Formulators are obtained using different numbers of expansion terms. Taking into account that the Headword/Compound Headword represents the focus of the query and that the expansion terms are exactly the same for both formulators, these results empathize, once again, the impact that different query formulation strategies have on the retrieval performance.

5.3.4 Headword/Compound Headword Expansion Lucene Results

Table 5.5 presents the Headword/Compound Headword expansion best results when Lucene is used as search engine, as well as the results obtained by previously evaluated formulators for comparative purposes. All the results of the Headword/Compound Headword expansion when using Lucene as search engine are detailed in Section E.2 of Appendix E.

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator	102 (55,7%)	0.19	0.33
Regex Query Formulator	103 (56.3%)	0.19	0.33
Query Reformulation Formulator with Keyword Query Formulator	108 (59%)	0.23	0.39
Query Reformulation Formulator with Regex Query Formulator	109 (60%)	0.22	0.37
Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**111 (60.6%)**	**0.19**	**0.32**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**106 (57.9%)**	**0.2**	**0.32**

Table 5.5: Headword/Compound Headword best expansion results when using Lucene search engine.

It can be seen that Headword/Compound Headword best expansion result when using the Regex Query Formulator does not improve at all the best results obtained so far using the Query Reformulation Formulator with both the Keyword and Regex Query Formulators. As for the Headword/Compound Headword best expansion result when using the Keyword Query Formulator, it can be seen a moderate and significant decrease of 0.04 and 0.08, respectively, over the best values of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ obtained by the Query Reformulation Formulator. Although there is an improvement of two questions (0,06%) over the #Question$_{1+PosPassage}$ obtained by the Query Reformulation Formulator, due to the aforementioned discrepancies in the MRR values we consider the latter formulator to have a better retrieval performance than the Query Expansion Formulator using the Keyword

Query Formulator.

When considering the improvements over the Keyword Query Formulator, we see that there is an increase of nine questions (4.9%) in the #Question$_{1+PosPassage}$ despite a slight decrease of 0.01 in the MRR$_{QPOSPASSAGES}$ value. Regarding the improvements over the Regex Query Formulator, despite a slight decrease of 0.01 in the MRR$_{QPOSPASSAGES}$ value, it can be seen an increase of four questions (2.2%) in the #Question$_{1+PosPassage}$ as well as a slight increase of 0.01 in the MRR$_{ALLQ}$ value.

As opposed of what was verified when using the Bing search, the Headword/Compound Headword best expansion results, when using both Keyword and Regex Query Formulators, are obtained when all the expansion terms are used, which seems to indicate that the relevance of certain expansion terms varies according to the search engine used to process the expanded query.

5.3.5 Main Verbs Expansion Bing Results

Table 5.6 presents the Main Verbs expansion best results when Bing is used as search engine, as well as the results obtained by previously evaluated formulators for comparative purposes. All the results of the Main Verbs expansion when using Bing as search engine are detailed in Section E.3 of Appendix E.

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator	137 (74.9%)	0.37	0.5
Regex Query Formulator	137 (74.9%)	0.34	0.45
Query Reformulation Formulator with Keyword Query Formulator	143 (78.1%)	0.45	0.57
Query Reformulation Formulator with Regex Query Formulator	143 (78.1%)	0.44	0.57
Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**150 (81.9%)**	**0.44**	**0.53**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**140 (76.5%)**	**0.37**	**0.48**

Table 5.6: Main Verbs best expansion results when using Bing search engine.

Before performing a detailed analysis of the presented results, it is important mention that the Main Verbs best expansion results surpass the best Headword/Compound Headword expansion results presented in Section 5.3.3. The biggest improvements are verified for the #Question$_{1+PosPassage}$ with a discrepancy of 10 positive questions (5.4%) when expanding queries with the Keyword Query Formulator, and a discrepancy of 11 positive questions (6.1%) when expanding queries with the Regex Query Formulator. Thus, we can conclude that when considering the individual contributions for the retrieval

process of expanding the Question Headword/Compound Headword and the Main Verbs of a query, the latter stands out as the best individual approach to use with Bing search engine (until this moment).

Focusing now only on the presented results, it can be seen that Main Verbs best expansion result when using the Regex Query Formulator does not improve at all the best results obtained so far using the Query Reformulation Formulator with both the Keyword and Regex Query Formulators. As for the Main Verbs best expansion result when using the Keyword Query Formulator, it can be seen a significant increase of seven questions (3.8%) over the #Question$_{1+PosPassage}$ obtained by the Query Reformulation Formulator, as well as a moderate decrease of 0.04 in the MRR$_{QPOSPASSAGES}$ value. We consider the increase of the #Question$_{1+PosPassage}$ to be of more relevance when compared to the observed decrease in the MRR$_{QPOSPASSAGES}$ value since it results, in the best case scenario, in seven more correct questions answered. Thus, we consider that the Main Verbs best expansion result when using the Keyword Query Formulator attains a better retrieval performance than the results obtained with the Query Reformulation Formulator.

Finally, it can be seen that the Main Verbs best expansion results improves the results obtained by the Keyword and Regex Query Formulators.

5.3.6 Main Verbs Expansion Lucene Results

Table 5.7 presents the Main Verbs expansion best results when Lucene is used as search engine, as well as the results obtained by previously evaluated formulators for comparative purposes. All the results of the Main Verbs expansion when using Lucene as search engine are detailed in Section E.4 of Appendix E.

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator	102 (55.7%)	0.19	0.33
Regex Query Formulator	103 (56.3%)	0.19	0.33
Query Reformulation Formulator with Keyword Query Formulator	108 (59%)	0.23	0.39
Query Reformulation Formulator with Regex Query Formulator	109 (60%)	0.22	0.37
Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**110 (60.1%)**	**0.19**	**0.31**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**102 (55.7%)**	**0.17**	**0.3**

Table 5.7: Main Verbs best expansion results when using Lucene search engine.

As opposed of what was verified for Bing search engine, where the Main Verbs expansion ap-

proach yields better results than the respective Headword/Compound Headword expansion results, for Lucene the latter expansion approach yields the best retrieval performance over the Main Verbs results presented in this section. Therefore, until this moment the Main Verbs expansion is the best individual approach to use with Bing search engine and the Headword/Compound Headword expansion stands as the best individual approach to use with Lucene search engine.

Focusing now only on the presented results, it can be seen that Main Verbs best expansion result when using the Regex Query Formulator does not improve at all the best results obtained so far using the Query Reformulation Formulator with both the Keyword and Regex Query Formulators. As for the Main Verbs best expansion result when using the Keyword Query Formulator, it can be seen a moderate and significant decrease of 0.04 and 0.08, respectively, over the best values of MRR_{ALLQ} and $\text{MRR}_{QPOSPASSAGES}$ obtained by the Query Reformulation Formulator. Although there is an improvement of two questions (1.1%) over the $\#\text{Question}_{1+PosPassage}$ obtained by the Query Reformulation Formulator, due to the aforementioned discrepancies in the MRR values we consider the latter formulator to have a better retrieval performance than the Query Expansion Formulator using the Keyword Query Formulator.

When considering the improvements over the Keyword Query Formulator we see that there is an increase of eight questions (4.4%) in the $\#\text{Question}_{1+PosPassage}$ despite a slight decrease of 0.02 in the $\text{MRR}_{QPOSPASSAGES}$ value. On the other hand, no improvements were attained over the results obtained by the Regex Query Formulator.

5.3.7 Headword/Compound Headword and Main Verbs Simultaneous Expansion Bing Results

Table 5.8 presents the Headword/Compound Headword and Main Verbs expansion best results when Bing is used as search engine, as well as the results obtained by previously evaluated formulators for comparative purposes. All the results of the Headword/Compound Headword and Main Verbs expansion when using Bing as search engine are detailed in Section E.5 of Appendix E. Regarding this evaluation, when a percentage such as 100% appears in the column of "Number of Expansion Terms" in the following table, that means that all the Headword/Compound Headword expansion terms retrieved are used as well as all the retrieved Main Verbs expansion terms. Also, to avoid using "Headword/Compound Headword and Main Verbs simultaneous expansion" each time we want to refer to this expansion approach, it will be referred from now on as "HeadVerb expansion".

Before performing a detailed analysis of the presented results, it is important to compare the Main Verbs best expansions results (the best query expansion results obtained so far for Bing search engine) with the HeadVerb expansion. For the HeadVerb best expansion result when using the Keyword Query

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator	137(74.9%)	0.37	0.5
Regex Query Formulator	137 (74.9%)	0.34	0.45
Query Reformulation Formulator with Keyword Query Formulator	143 (78.1%)	0.45	0.57
Query Reformulation Formulator with Regex Query Formulator	143 (78.1%)	0.44	0.57
Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**147 (80.3%)**	**0.45**	**0.56**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**138 (75.4%)**	**0.38**	**0.5**

Table 5.8: Headword/Compound Headword and Main Verbs best expansion results when using Bing search engine.

Formulator, it can be seen a slight and moderate decrease of 0.01 and 0.03, respectively, over the values of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ obtained by the Main Verbs best expansion result when using the Keyword Query Formulator. However, there is an improvement of three questions (1.6%) over the #Question$_{1+PosPassage}$ obtained by the Main Verbs best expansion result when using the Keyword Query Formulator. We consider the increase of the #Question$_{1+PosPassage}$ to be of more relevance when compared to the observed decreased in the MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ values since it results, in the best case scenario, in three more correct questions answered. Thus, we consider that the Main Verbs expansion when using the Keyword Query Formulator attains a better retrieval performance than the HeadVerb expansion also using the Keyword Query Formulator. Using the same train of thought, the same consideration is achieved regarding Main Verbs expansion when using the Regex Query Formulator and HeadVerb expansion also using that formulator, i.e, Main Verbs expansion when using the Regex Query Formulator attains a better retrieval performance than the HeadVerb expansion also using the Regex Query Formulator. Thus, we can conclude that the Main verbs expansion approach stands out as the most beneficial expansion approach for the retrieval task using Bing search engine.

Focusing now only on the presented results, it can be seen that HeadVerb best expansion result when using the Regex Query Formulator does not improve at all the best results obtained so far using the Query Reformulation Formulator with both the Keyword and Regex Query Formulators. As for the HeadVerb best expansion result when using the Keyword Query Formulator, it can be seen a slight decrease of 0.01 over the value of MRR$_{QPOSPASSAGES}$ obtained by the Query Reformulation Formulator, as well as an improvement of four questions questions (2.2%) over the #Question$_{1+PosPassage}$. Since the increase of the #Question$_{1+PosPassage}$ is of more relevance when compared to the observed decreased in the MRR$_{QPOSPASSAGES}$ value since it results, in the best case scenario, in three more correct questions

answered, we consider the HeadVerb expansion approach when using the Keyword Query Formulator to have a better retrieval performance than the Query Reformulation Formulator. The HeadVerb best expansion results also improves the results obtained by the Keyword and Regex Query Formulators.

From all the analysis of the different expansion approaches using Bing search engine, we can conclude that the Main verbs expansion approach stands out as the most beneficial expansion approach, having its best result when using the Keyword Query Formulator to generate the queries to be expanded as well as all the retrieved expansion terms.

5.3.8 Headword/Compound Headword and Main Verbs Simultaneous Expansion Lucene Results

Table 5.9 presents the Headword/Compound Headword and Main Verbs expansion best results when Lucene is used as search engine, as well as the results obtained by previously evaluated formulators for comparative purposes. All the results of the Headword/Compound Headword and Main Verbs expansion when using Lucene as search engine are detailed in Section E.6 of Appendix E.

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator	102(55.7%)	0.19	0.33
Regex Query Formulator	103 (56.3%)	0.19	0.33
Query Reformulation Formulator with Keyword Query Formulator	108 (59%)	0.23	0.39
Query Reformulation Formulator with Regex Query Formulator	109 (60%)	0.22	0.37
Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**115 (62.8%)**	**0.19**	**0.3**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
100%	**108 (59%)**	**0.19**	**0.32**

Table 5.9: Headword/Compound Headword and Main Verbs best expansion results when using Lucene search engine.

Before performing a detailed analysis of the presented results, it is important to compare the Headword/Compound Headword best expansions results (the best query expansion results obtained so far for Lucene search engine) with the HeadVerb expansion. For the HeadVerb best expansion result when using the Keyword Query Formulator, it can be seen a slight decrease of 0.02 over the value of MRR$_{QPOSPASSAGES}$ obtained by the Headword/Compound Headword best expansion result when using the Keyword Query Formulator. However, there is an improvement of four questions (2.2%) over the #Question$_{1+PosPassage}$ obtained by the Headword/Compound Headword best expansion re-

sult when using the Keyword Query Formulator. We consider the increase of the #Question$_{1+PosPassage}$ to be of more relevance when compared to the observed decreased in the MRR$_{QPOSPASSAGES}$ value since it results, in the best case scenario, in four more correct questions answered. Thus, we consider that the HeadVerb expansion when using the Keyword Query Formulator attains a better retrieval performance than the Headword/Compound Headword expansion also using the Keyword Query Formulator. Using the same train of thought, the same consideration is achieved regarding HeadVerb expansion when using the Regex Query Formulator and Headword/Compound Headword expansion also using that formulator, i.e, HeadVerb expansion when using the Regex Query Formulator attains a better retrieval performance than the Headword/Compound Headword expansion also using the Regex Query Formulator. Thus, we can conclude that the HeadVerb expansion approach stands out as the most beneficial expansion approach for the retrieval task using Lucene search engine.

Focusing now only on the presented results, it can be seen that HeadVerb best expansion result when using the Regex Query Formulator does not improve at all the best results obtained so far using the Query Reformulation Formulator with both the Keyword and Regex Query Formulators. As for the HeadVerb best expansion result when using the Keyword Query Formulator, it can be seen a moderate and significant decrease of 0.03 and 0.09, respectively, over the best values of MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ obtained by the Query Reformulation Formulator. There is also an improvement of seven questions (3.8%) over the #Question$_{1+PosPassage}$ obtained by the Query Reformulation Formulator. Although the increase of the #Question$_{1+PosPassage}$ is significant, since it can lead, in the best case scenario, to three more correct questions answered, the observed decrease in the MRR$_{ALLQ}$ and MRR$_{QPOSPASSAGES}$ values (being the last one the most important), are also significant. Therefore, we cannot say that the Query Reformulation Formulator have a better retrieval performance the HeadVerb best expansion result when using the Keyword Query Formulator, and vice versa. As a last remark regarding the results, the HeadVerb best expansion results also improves the results obtained by the Keyword and Regex Query Formulators.

From all the analysis of the different expansion approaches using Lucene search engine, we can conclude that the Question Reformulation and HeadVerb expansion approach stand out as the most beneficial approaches when using Lucene.

5.4 Query Reformulation and Expansion Combination

Sections 5.2 and 5.3 described the implemented query reformulation and query expansion formulators and depicted the improvements attained by Just.Ask's Passage Retrieval module when using each one of the aforementioned formulators. In this section we utilize both formulators simultaneously to ascertain if improvements over the retrieval performance of each formulator can be achieved by doing so.

5.4.1 Experimental Setup

Using the evaluation setting and evaluation metrics defined in Section 3.2.1, the Passage Retrieval module was tested using both Query Reformulation and Query Expansion Formulators simultaneously. Thus, all the possible combinations between the best parameterizations of each formulation method for Bing and Lucene search engines, respectively, were combined with that purpose. More specifically, the following combinations were used for Bing and Lucene search engines:

- 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.
- 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.

5.4.2 Query Reformulation and Expansion Combination Bing Results

Table 5.10 presents the Query Reformulation and Expansion combination results when Bing is used as search engine, as well as the best results obtained individually by each one of those formulators and Just.Ask's baseline retrieval results for comparative purposes.

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 #Question$_{1+PosPassage}$ where one more question with at least one positive passage is retrieved by Bing Combination 1 over the #Question$_{1+PosPassage}$ 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 #Question$_{1+PosPassage}$, 0.15 in the MRR$_{ALLQ}$ and 0.11 in the MRR$_{QPOSPASSAGES}$.

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator (Baseline)	137 (74.9%)	0.37	0.5
Query Reformulation Formulator with Keyword Query Formulator	143 (78.1%)	0.45	0.57
Query Reformulation Formulator with Regex Query Formulator	143 (78.1%)	0.44	0.57
Query Expansion Formulator with Keyword Query Formulator	150 (81.9%)	0.44	0.53
Query Reformulation and Expansion Formulators Combinations - 50 passages			
Parameterizations Combinations	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Bing Combination 1	**154 (84.1%)**	**0.52**	**0.61**
Bing Combination 2	153 (83.6%)	0.52	0.61

Table 5.10: Query Reformulation and Expansion combination results when using Bing search engine.

5.4.3 Query Reformulation and Expansion Combination Lucene Results

Table 5.11 presents the Query Reformulation and Expansion combination results when Lucene is used as search engine, as well as the best results obtained individually by each one of those formulators and Just.Ask's baseline retrieval results for comparative purposes.

Other Formulators - 50 passages			
Formulator	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Keyword Query Formulator (Baseline)	102 (55.7%)	0.19	0.33
Query Reformulation Formulator with Keyword Query Formulator	108 (59%)	0.23	0.39
Query Reformulation Formulator with Regex Query Formulator	109 (60%)	0.22	0.37
Query Expansion Formulator with Keyword Query Formulator	115 (62.8%)	0.19	0.3
Query Reformulation and Expansion Formulators Combinations - 50 passages			
Parameterizations Combinations	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
Lucene Combination 1	**122 (66.6%)**	**0.22**	**0.33**
Lucene Combination 2	**122 (66.6%)**	**0.22**	**0.33**

Table 5.11: 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 and Query Expansion Formulators are identical for all evaluation metrics, and they only surpass the best results attained by the other formulators in the #Question$_{1+PosPassage}$ evaluation metric. Specifically, there is an improvement of seven questions (3.8%) over the best value of #Question$_{1+PosPassage}$ 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 #Question$_{1+PosPassage}$ 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 #Question$_{1+PosPassage}$ and 0.03 in the MRR$_{ALLQ}$.

6 Conclusions and Future Work

This last chapter presents the final remarks of this thesis, presenting the main contributions and summarizing the work that was accomplished. It concludes by presenting some ideas for future work.

6.1 Final Remarks

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. 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.

An experiment utilizing ensemble query classifiers built from several SVM, RBF and Naive-Bayes query classifiers, whose classifications were combined using different voting methods, was conducted to improve the previous results, although no improvements were achieved. 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 formulations that questions possess. Results obtained with this query formulator improved the baseline retrieval results of Just.Ask for both of the search engines used.

The Query Expansion Formulator 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. The evaluation of this query formulator led to some conclusions: the individual and conjugated contributions to the retrieval performance of expanding the question headword/compound headword and the main verbs of a query varies according to the search engine used, and using all the retrieved expansion terms usually leads to best retrieval performance. Results obtained with this query formulator improved the baseline retrieval results of Just.Ask for both of the search engines used.

Finally, the use of both of the previous query formulators 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$_{QPOSPASSAGES}$. 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$_{ALLQ}$.

6.2 Main Contributions

The main contributions of this thesis are the following:

- Adaptation of the train and test datasets used in the Question Classification task for the Query Classification task;
- Training and testing of a SVM Query Classifier using using features developed for the Query Classification task that take into consideration the main characteristics of queries;
- Creation of ensemble query classifiers built from several SVM, RBF and Naive-Bayes query classifiers;
- Creation of methods to perform simultaneous Query and Question Classification;

- Creation of a query formulator that 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.
- Creation of a query formulator that expands the question headword/compound headword and the main verbs that exist in the user question with semantically related terms obtained using Wordnet.

6.3 Future Work

In future work it would be interesting to:

- Perform multi-label query classification, i.e, a query can have multiple classifications instead of just one classification as it was considered in this work. It is expected for this approach to improve the overall Just.Ask performance, since the use of multiple classifications may result in more candidate answers extracted from obtained passages and consecutively in more correct answers;
- Use the explicit hierarchy in the Li and Roth taxonomy to guide the classification process instead of using each hierarchy in a flat manner. For example, we can classify a query using both coarse and fine-grained classification. If the fine-grained category of that query does not belong to the coarse-grained category, we should define the coarse-grained category as the final category of that query. The rationale for this decision is based on the fact that coarse-grained classification achieves an higher accuracy when compared to fine-grained classification;
- Study methods to successfully identify in a query the headword, named entities and POS tags, so that they can be used as features for the Query Classification task in their full extent;
- Improve the created ensemble query classifiers by adding new query classifier types such as Neural Networks, K-Nearest Neighbors, Decision Trees and by devising more robust voting methods;
- 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 Reformulation 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[1].

[1] http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/

Bibliography

Billerbeck, B., & Zobel, J. (2004). Techniques for efficient query expansion. In *Proc. string processing and information retrieval symp* (pp. 30–42). Springer-Verlag.

Bomhoff, M. J., Huibers, T. W. C., & van der Vet, P. E. (2005). User intentions in information retrieval. In R. van Zwol (Ed.), (pp. 47–54). Utrecht: Universiteit Utrecht.

Cardoso, N., Silva, M., & Martins, B. (2007). The university of lisbon at clef 2006 ad-hoc task. In *Evaluation of multilingual and multi-modal information retrieval* (p. 51-56). Springer Berlin / Heidelberg.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37–46.

Collins, M. J. (1999). *Head-driven statistical models for natural language parsing*. Unpublished doctoral dissertation, Philadelphia, PA, USA. (Supervisor-Marcus, Mitchell P.)

Crestani, F., Lalmas, M., Van Rijsbergen, C. J., & Campbell, I. (1998). "is this document relevant?...probably": a survey of probabilistic models in information retrieval. *ACM Comput. Surv.*, *30*, 528–552.

Croft, B. W., & Lafferty, J. (2003). *Language modeling for information retrieval (the information retrieval series)*. Springer.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Science*, *41*(6), 391–407.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Proceedings of the first international workshop on multiple classifier systems* (pp. 1–15). London, UK: Springer-Verlag.

Efthimiadis, E. N. (1993). A user-centred evaluation of ranking algorithms for interactive query expansion. In *Proceedings of the 16th annual international acm sigir conference on research and development in information retrieval* (pp. 146–159). New York, NY, USA: ACM.

Efthimiadis, E. N. (1995). User choices: a new yardstick for the evaluation of ranking algorithms for interactive query expansion. *Inf. Process. Manage.*, *31*, 605–620.

Fellbaum, C. (Ed.). (1998). *WordNet: An electronic lexical database*. MIT Press.

Fialho, P., Curto, S., Mendes, A., & Conehur, L. (2011, September). *Wordnet framework improvements for nlp: Defining abstraction and scalability layers* (Tech. Rep. No. 8113). Technical University of Lisbon - Spoken Language Systems Lab.

Fleiss, J. L., Cohen, J., & Everitt, B. S. (1969). Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, *72*(5), 323–327.

Gravano, L., Hatzivassiloglou, V., & Lichtenstein, R. (2003). Categorizing web queries according to geographical locality. In *Proceedings of the twelfth international conference on information and knowledge management* (pp. 325–333). New York, NY, USA: ACM.

Hammo, B., Abu-Salem, H., & Lytinen, S. (2002). Qarab: a question answering system to support the arabic language. In *Proceedings of the acl-02 workshop on computational approaches to semitic languages* (pp. 1–11). Stroudsburg, PA, USA: Association for Computational Linguistics.

Hsu, M.-H., Tsai, M.-F., & Chen, H.-H. (2008). Combining wordnet and conceptnet for automatic query expansion: a learning approach. In *Proceedings of the 4th asia information retrieval conference on information retrieval technology* (pp. 213–224). Berlin, Heidelberg: Springer-Verlag.

Judge, J., Cahill, A., & Genabith, J. van. (2006). Questionbank: creating a corpus of parse-annotated questions. In *Acl-44: Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics* (pp. 497–504). Morristown, NJ, USA: Association for Computational Linguistics.

Jurafsky, D., & Martin, J. H. (2008). *Speech and language processing (2nd edition) (prentice hall series in artificial intelligence)* (2 ed.). Prentice Hall.

Kang, I.-H., & Kim, G. (2003). Query type classification for web document retrieval. In *Proceedings of the 26th annual international acm sigir conference on research and development in informaion retrieval* (pp. 64–71). New York, NY, USA: ACM.

Kim, H., Kim, K., Lee, G. G., & Seo, J. (2001). Maya: a fast question-answering system based on a predictive answer indexer. In *Proceedings of the workshop on open-domain question answering - volume 12* (pp. 1–8). Stroudsburg, PA, USA: Association for Computational Linguistics.

Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on computational linguistics* (pp. 1–7). Morristown, NJ, USA: Association for Computational Linguistics.

Lin, J. (2007, April). An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Inf. Syst.*, *25*, 1-15.

Lv, Y., Zhai, C., & Chen, W. (2011). A boosting approach to improving pseudo-relevance feedback. In *Proceedings of the 34th international acm sigir conference on research and development in information* (pp. 165–174). New York, NY, USA: ACM.

Petrov, S., & Klein, D. (2007, April). Improved inference for unlexicalized parsing. In *Human language technologies 2007: The conference of the north american chapter of the association for computational linguistics; proceedings of the main conference* (pp. 404–411). Rochester, New York: Association for Computational Linguistics.

Pires, R. (2012). A common evaluation setting for just.ask, open ephyra and aranea qa systems. *abs/1205.1779*. Paper submitted on arXiv.org.

Rocchio, R. S., J. J. (1971). The smart retrieval system - experiments in automatic document processing. In (pp. 313–323). Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Ruthven, I., & Lalmas, M. (2003, June). A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, *18*(2), 95–145.

Sakai, T., Saito, Y., Ichimura, Y., Koyama, M., Kokubu, T., & Manabe, T. (2004). Askmi: A japanese question answering system based on semantic role analysis. In *In proceedings of riao 2004* (pp. 215–231).

Sarmento, L. (2008). Experiments with query expansion in the RAPOSA (FOX) question answering system. *Communications of the ACM*, *8*, 792–798.

Schlaefer, N., Gieselman, P., & Sautter, G. (2006). The ephyra qa system at trec 2006. In *In proceedings of the fifteenth text retrieval conference (trec)* (pp. 2–4).

Schlaefer, N., Gieselmann, P., Schaaf, T., & Waibe, A. (2006). A pattern learning approach to question answering within the ephyra framework. In *Text, speech and dialogue* (p. 687-694). Springer Berlin / Heidelberg.

Schlaefer, N., Ko, J., Betteridge, J., Pathak, M. A., Nyberg, E., & Sautter, G. (2007). Semantic extensions of the ephyra qa system for trec 2007. In *In proceedings of trec. 2007* (p. 4-5).

Shen, D., Pan, R., Sun, J.-T., Pan, J. J., Wu, K., Yin, J., et al. (2005, December). Q2c@ust: our winning solution to query classification in kddcup 2005. *SIGKDD Explor. Newsl.*, *7*, 100–110.

Silva, J., Coheur, L., & Mendes, A. (n.d.). *Just.ask - a multi-pronged approach to question answering*. Paper submitted to the Internacional Journal on Artificial Intelligence Tools in October 2010.

Silva, J., Coheur, L., Mendes, A., & Wichert, A. (2010). From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*.

Sun, R., Jiang, J., Tan, Y. F., Cui, H., Chua, T. seng, & Kan, M. yen. (2005). Using syntactic and semantic relation analysis in question answering. In *Text retrieval conference* (pp. -1–1).

Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international acm sigir conference on research and development in information retrieval* (pp. 61–69). New York, NY, USA: Springer-Verlag New York, Inc.

Xu, J., & Croft, W. B. (2000). Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, *18*, 79–112.

Xu, Y., Jones, G. J., & Wang, B. (2009). Query dependent pseudo-relevance feedback based on wikipedia. In *Proceedings of the 32nd international acm sigir conference on research and development in information retrieval* (pp. 59–66). New York, NY, USA: ACM.

Zhang, J., Deng, B., & Li, X. (2009). Concept based query expansion using wordnet. In *Proceedings of the 2009 international e-conference on advanced science and technology* (pp. 52–55). Washington, DC, USA: IEEE Computer Society.

Zheng, Z. (2002). Answerbus question answering system. In *Proceedings of the second international conference on human language technology research* (pp. 399–404). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

I Appendices

A Regex Method Demonstration

ABBREVIATION_ABBREVIATION What is the abbreviation of the National Bureau of Investigation ?
ABBREVIATION_ABBREVIATION National Bureau of Investigation abbreviation

ABBREVIATION_EXPANSION What does CPR stand for ?
ABBREVIATION_EXPANSION CPR stand for

DESCRIPTION_DEFINITION What are “inkhorn terms” ?
DESCRIPTION_DEFINITION “inkhorn terms”

DESCRIPTION_DESCRIPTION What does Lloyd ’s Lutine Bell announce ?
DESCRIPTION_DESCRIPTION Lloyd Lutine Bell announce

DESCRIPTION_MANNER How can I find out my biorhythm ?
DESCRIPTION_MANNER how to find out my biorhythm

DESCRIPTION_REASON What are the most common causes of death in the U.S. ?
DESCRIPTION_REASON most common causes of death in the U.S.

ENTITY_ANIMAL What kind of dog is Scooby Doo ?
ENTITY_ANIMAL kind of dog is Scooby Doo

ENTITY_BODY What is the weakest bone in the body ?
ENTITY_BODY weakest bone in the body

ENTITY_COLOR What color is Mr. Spock ’s blood ?
ENTITY_COLOR Mr. Spock blood color

ENTITY_CREATIVE What film or films has Jude Law appeared in ?
ENTITY_CREATIVE film or films has Jude Law appeared in

ENTITY_EVENT What was the longest war in U.S. history ?
ENTITY_EVENT longest war in U.S. history

ENTITY_FOOD What foods contain the most protein ?
ENTITY_FOOD foods contain the most protein

ENTITY_INSTRUMENT What instrument is Ray Charles best known for playing ?
ENTITY_INSTRUMENT Ray Charles instrument

ENTITY_LANGUAGE What is the primary language of the Philippines ?
ENTITY_LANGUAGE the Philippines language

ENTITY_LETTER What letter adorns the flag of Rwanda ?
ENTITY_LETTER letter adorns the flag of Rwanda

ENTITY_MEDICINE What cancer is commonly associated with AIDS ?
ENTITY_MEDICINE cancer is commonly associated with AIDS

ENTITY_OTHER What are the seven wonders of the world ?
ENTITY_OTHER seven wonders of the world

ENTITY_PLANT What plant has the largest seed ?
ENTITY_PLANT plant has the largest seed

ENTITY_PRODUCT What kind of hat does Bat Masterson wear ?
ENTITY_PRODUCT kind of hat does Bat Masterson wear

ENTITY_RELIGION What Caribbean cult did Marcus Garvey originate ?
ENTITY_RELIGION Caribbean cult did Marcus Garvey originate

ENTITY_SPORT What game does Garry Kasparov play ?
ENTITY_SPORT game does Garry Kasparov play

ENTITY_SUBSTANCE What is the second hardest substance ?
ENTITY_SUBSTANCE second hardest substance

ENTITY_SYMBOL What playing card symbolizes death ?
ENTITY_SYMBOL playing card symbolizes death

ENTITY_TECHNIQUE What are the different approaches of systems analysis ?
ENTITY_TECHNIQUE different approaches of systems analysis

ENTITY_TERM How do you say 2 in Latin ?
ENTITY_TERM how to say 2 in Latin

ENTITY_VEHICLE What was the name of Robert Fulton 's most famous steamboat ?
ENTITY_VEHICLE name of Robert Fulton most famous steamboat

ENTITY_WORD What is the plural of gulf ?
ENTITY_WORD plural of gulf

HUMAN_DESCRIPTION Who is Nicolo Paganini ?
HUMAN_DESCRIPTION Nicolo Paganini

HUMAN_GROUP What group kidnaped Patricia Hearst ?
HUMAN_GROUP kidnaped Patricia Hearst

HUMAN_INDIVIDUAL What are the top boy names in the U.S. ?
HUMAN_INDIVIDUAL top boy names in the U.S.

HUMAN_TITLE What was Queen Victoria 's title regarding India ?
HUMAN_TITLE Queen Victoria title regarding India

LOCATION_CITY What city did the Flintstones live in ?
LOCATION_CITY city Flintstones live in

LOCATION_COUNTRY What countries earn the most from tourism ?
LOCATION_COUNTRY countries earn the most from tourism

LOCATION_MOUNTAIN What is New England 's highest mountain ?
LOCATION_MOUNTAIN New England highest mountain

LOCATION_OTHER What airport is on the Piccadilly subway line ?
LOCATION_OTHER airport is on the Piccadilly subway line

LOCATION_STATE What New England state covers 5.9 square miles ?
LOCATION_STATE New England state covers 5.9 square miles

NUMERIC_CODE What is the telephone number for the University of Kentucky ?
NUMERIC_CODE telephone number for the University of Kentucky

NUMERIC_COUNT How many Beatles ' records went #1 ?
NUMERIC_COUNT number of Beatles records went #1

NUMERIC_DATE What date is Boxing Day ?
NUMERIC_DATE Boxing Day date

NUMERIC_DISTANCE How far is London UK from California ?
NUMERIC_DISTANCE distance from London UK to California

NUMERIC_MONEY How much did the first Barbie doll sell for in 1959 ?
NUMERIC_MONEY first Barbie doll sell for in 1959

NUMERIC_OTHER What is the population of Kansas ?
NUMERIC_OTHER population of Kansas

NUMERIC_PERCENT What percentage of the body is muscle ?
NUMERIC_PERCENT percentage of the body is muscle

NUMERIC_PERIOD How old is the universe ?
NUMERIC_PERIOD universe age

NUMERIC_SIZE How big is Australia ?
NUMERIC_SIZE big is Australia

NUMERIC_SPEED How fast is light
NUMERIC_SPEED fast is light

NUMERIC_TEMPERATURE What is the temperature today ?
NUMERIC_TEMPERATURE temperature today

NUMERIC_WEIGHT What do Englishmen weigh themselves in ?
NUMERIC_WEIGHT Englishmen weigh themselves in

B SVM Classifier Feature Combination Results

This appendix shows, for both coarse and fine-grained classification, the accuracy of the feature combinations performed by the algorithm described in Section 4.2.4, using the SVM classifier. For each type of classification, the results are presented for both Just and Regex methods, according to the iteration that generated those results. The results of an iteration that does not improve the results of the last iteration are not shown due to their lack of relevance.

B.1 Coarse-Grained Classification

B.1.1 First Iteration

Accuracy for the Just Method	
Features	Just Method
Binary Word Shape	36.4%
Word Shape	38.2%
Binary Verb	57.2%
Semantic Headword	46.4%
First Token	64%
Binary Length	30.2%
Ner-Increment	31.2%
Ner-Replace	**74%**
Number of Chars	33.6%
Number of Tokens	31.8%
Phrase	33.2%
POS tags	38.2%
Pseudo Headword	43.4%
Unigrams	**71.8%**
B. Unigrams	**72%**
Bigrams	33%
B. Bigrams	33%

Accuracy for the Regex Method	
Features	Just Method
Binary Word Shape	35.6%
Word Shape	34.8%
Binary Verb	56.8%
Semantic Headword	54.2%
First Token	48%
Binary Length	33%
Ner-Increment	34.4%
Ner-Replace	**83.8%**
Number of Chars	32.2%
Number of Tokens	30.4%
Phrase	37.6%
POS tags	39.4%
Pseudo Headword	48.8%
Unigrams	**83.8%**
B. Unigrams	**84.2%**
Bigrams	59.2%
B. Bigrams	59.2%

B.1.2 Second Iteration

Accuracy for the Just Method	
Features	Just Method
Ner-Replace + Binary Word Shape	**75.2%**
Ner-Replace + Word Shape	73.4%
Ner-Replace + Binary Verb	74%
Ner-Replace + Semantic Headword	74.8%
Ner-Replace + First Token	74.8%
Ner-Replace + Binary Length	73%
Ner-Replace + Ner-Increment	73.8%
Ner-Replace + Number of Chars	72%
Ner-Replace + Number of Tokens	73.6%
Ner-Replace + Phrase	74.4%
Ner-Replace + POS tags	74.2%
Ner-Replace + Pseudo Headword	73%
Unigrams + Binary Word Shape	**75.2%**
Unigrams + Word Shape	72.8%
Unigrams + Binary Verb	74%
Unigrams + Semantic Headword	73.2%
Unigrams + First Token	74.6%
Unigrams + Binary Length	72%
Unigrams + Ner-Increment	72.4%
Unigrams + Ner-Replace	71.8%
Unigrams + Number of Chars	72.8%
Unigrams + Number of Tokens	72.6%
Unigrams + Phrase	74.2%
Unigrams + POS tags	72.8%
Unigrams + Pseudo Headword	71.8%
B. Unigrams + Binary Word Shape	74.6%
B. Unigrams + Word Shape	72.2%
B. Unigrams + Binary Verb	73.6%
B. Unigrams + Semantic Headword	73.6%
B. Unigrams + First Token	**75%**
B. Unigrams + Binary Length	71.8%
B. Unigrams + Ner-Increment	72%
B. Unigrams + Ner-Replace	72.2%
B. Unigrams + Number of Chars	72.2%
B. Unigrams + Number of Tokens	71.8%
B. Unigrams + Phrase	73.4%
B. Unigrams + POS tags	72.4%
B. Unigrams + Pseudo Headword	72%

Accuracy for the Regex Method	
Features	Just Method
Ner-Replace + Binary Word Shape	83.8%
Ner-Replace + Word Shape	83.2%
Ner-Replace + Binary Verb	83.6%
Ner-Replace + Semantic Headword	84.4%
Ner-Replace + First Token	**85.6%**
Ner-Replace + Binary Length	83.2%
Ner-Replace + Ner-Increment	83.8%
Ner-Replace + Number of Chars	83.4%
Ner-Replace + Number of Tokens	83.2%
Ner-Replace + Phrase	84%
Ner-Replace + POS tags	85.4%
Ner-Replace + Pseudo Headword	82.4%
Unigrams + Binary Word Shape	83%
Unigrams + Word Shape	84.2%
Unigrams + Binary Verb	84.4%
Unigrams + Semantic Headword	84%
Unigrams + First Token	**85.2%**
Unigrams + Binary Length	82.6%
Unigrams + Ner-Increment	83.2%
Unigrams + Ner-Replace	83.6%
Unigrams + Number of Chars	84.4%
Unigrams + Number of Tokens	83.6%
Unigrams + Phrase	84.2%
Unigrams + POS tags	85%
Unigrams + Pseudo Headword	81.8%
B. Unigrams + Binary Word Shape	82.8%
B. Unigrams + Word Shape	83.4%
B. Unigrams + Binary Verb	**85.2%**
B. Unigrams + Semantic Headword	83.6%
B. Unigrams + First Token	85%
B. Unigrams + Binary Length	84%
B. Unigrams + Ner-Increment	84.2%
B. Unigrams + Ner-Replace	84.6%
B. Unigrams + Number of Chars	84.4%
B. Unigrams + Number of Tokens	83.6%
B. Unigrams + Phrase	83.8%
B. Unigrams + POS tags	85%
B. Unigrams + Pseudo Headword	81.8%

B.1.3 Third Iteration

Accuracy for the Just Method	
Features	Just Method
Ner-Replace + Binary Word Shape + Wordshape	75.2%
Ner-Replace + Binary Word Shape + Binary Verb	76.2%
Ner-Replace + Binary Word Shape + Semantic Headword	75%
Ner-Replace + Binary Word Shape + First Token	**78.2%**
Ner-Replace + Binary Word Shape + Binary Length	75.2%
Ner-Replace + Binary Word Shape + Ner-Increment	75.2%
Ner-Replace + Binary Word Shape + Number of Chars	76.2%
Ner-Replace + Binary Word Shape + Number of Tokens	74.6%
Ner-Replace + Binary Word Shape + Phrase	75.6%
Ner-Replace + Binary Word Shape + POS tags	76.6%
Ner-Replace + Binary Word Shape + Pseudo Headword	75%
Unigrams + Binary Word Shape + Wordshape	74.4%
Unigrams + Binary Word Shape + Binary Verb	74.4%
Unigrams + Binary Word Shape + Semantic Headword	75%
Unigrams + Binary Word Shape + First Token	**78.6%**
Unigrams + Binary Word Shape + Binary Length	73.8%
Unigrams + Binary Word Shape + Ner-Increment	75%
Unigrams + Binary Word Shape + Ner-Replace	73.6%
Unigrams + Binary Word Shape + Number of Chars	75.2%
Unigrams + Binary Word Shape + Number of Tokens	74.8%
Unigrams + Binary Word Shape + Phrase	75.8%
Unigrams + Binary Word Shape + POS tags	75.8%
Unigrams + Binary Word Shape + Pseudo Headword	74%
B. Unigrams + First Token + Binary Word Shape	78%
B. Unigrams + First Token + Word Shape	75.8%
B. Unigrams + First Token + Binary Verb	75%
B. Unigrams + First Token + Semantic Headword	74.8%
B. Unigrams + First Token + Binary Length	75%
B. Unigrams + First Token + Ner-Increment	74%
B. Unigrams + First Token + Ner-Replace	73.4%
B. Unigrams + First Token + Number of Chars	75.4%
B. Unigrams + First Token + Number of Tokens	75.2%
B. Unigrams + First Token + Phrase	75.8%
B. Unigrams + First Token + POS tags	74.4%
B. Unigrams + First Token + Pseudo Headword	73.6%

Accuracy for the Regex Method	
Features	Just Method
Ner-Replace + First Token + Binary Word Shape	85.2%
Ner-Replace + First Token + Word Shape	84.4%
Ner-Replace + First Token + Binary Verb	85.2%
Ner-Replace + First Token + Semantic Headword	84.6%
Ner-Replace + First Token + Binary Length	85.4%
Ner-Replace + First Token + Ner-Increment	85.6%
Ner-Replace + First Token + Number of Chars	**85.8%**
Ner-Replace + First Token + Number of Tokens	85.4%
Ner-Replace + First Token + Phrase	85.8%
Ner-Replace + First Token + POS tags	85.4%
Ner-Replace + First Token + Pseudo Headword	84.2%
Unigrams + First Token + Binary Word Shape	84.6%
Unigrams + First Token + Word Shape	84.8%
Unigrams + First Token + Binary Verb	84.2%
Unigrams + First Token + Semantic Headword	85%
Unigrams + First Token + Binary Length	84.2%
Unigrams + First Token + Ner-Increment	85%
Unigrams + First Token + Ner-Replace	84.4%
Unigrams + First Token + Number of Chars	85.4%
Unigrams + First Token + Number of Tokens	85.2%
Unigrams + First Token + Phrase	84.4%
Unigrams + First Token + POS tags	**86%**
Unigrams + First Token + Pseudo Headword	84.6%
B. Unigrams + POS tags + Binary Word Shape	83.8%
B. Unigrams + POS tags + Word Shape	84.8%
B. Unigrams + POS tags + Binary Verb	85.2%
B. Unigrams + POS tags + Semantic Headword	84.8%
B. Unigrams + POS tags + First Token	85.4%
B. Unigrams + POS tags + Binary Length	84.8%
B. Unigrams + POS tags + Ner-Increment	84.2%
B. Unigrams + POS tags + Ner-Replace	84.4%
B. Unigrams + POS tags + Number of Chars	85%
B. Unigrams + POS tags + Number of Tokens	85.2%
B. Unigrams + POS tags + Phrase	85%
B. Unigrams + First Token + Pseudo Headword	82.8%

B.2 Fine-Grained Classification

B.2.1 First Iteration

Accuracy for the Just Method	
Features	Just Method
Binary Word Shape	33.6%
Word Shape	30.4%
Binary Verb	30.4%
Semantic Headword	34.2%
First Token	36.4%
Binary Length	11%
Ner-Increment	11%
Ner-Replace	**68.6%**
Number of Chars	32.4%
Number of Tokens	30.8%
Phrase	31.6%
POS tags	34%
Pseudo Headword	30.6%
Unigrams	**67.6%**
B. Unigrams	**67.6%**
Bigrams	23%
B. Bigrams	23%

Accuracy for the Regex Method	
Features	Just Method
Binary Word Shape	33.2%
Word Shape	31.8%
Binary Verb	50.4%
Semantic Headword	39.6%
First Token	34.6%
Binary Length	29.6%
Ner-Increment	11%
Ner-Replace	**78.8%**
Number of Chars	33.4%
Number of Tokens	32%
Phrase	33%
POS tags	38.6%
Pseudo Headword	40.6%
Unigrams	**78.4%**
B. Unigrams	**78.4%**
Bigrams	50%
B. Bigrams	50%

B.2.2 Second Iteration

Accuracy for the Just Method	
Features	Just Method
Ner-Replace + Binary Word Shape	**69.6%**
Ner-Replace + Word Shape	**69.6%**
Ner-Replace + Binary Verb	68.8%
Ner-Replace + Semantic Headword	68.2%
Ner-Replace + First Token	68.8%
Ner-Replace + Binary Length	68%
Ner-Replace + Ner-Increment	68.6%
Ner-Replace + Number of Chars	68.2%
Ner-Replace + Number of Tokens	69%
Ner-Replace + Phrase	67%
Ner-Replace + POS tags	**69.2%**
Ner-Replace + Pseudo Headword	65%
Unigrams + Binary Word Shape	**69.2%**
Unigrams + Word Shape	**69.2%**
Unigrams + Binary Verb	67.4%
Unigrams + Semantic Headword	66%
Unigrams + First Token	67.6%
Unigrams + Binary Length	67.6%
Unigrams + Ner-Increment	66.8%
Unigrams + Ner-Replace	66.4%
Unigrams + Number of Chars	67%
Unigrams + Number of Tokens	68.2%
Unigrams + Phrase	66.8%
Unigrams + POS tags	67.4%
Unigrams + Pseudo Headword	65%
B. Unigrams + Binary Word Shape	68.8%
B. Unigrams + Word Shape	**69.2%**
B. Unigrams + Binary Verb	67.4%
B. Unigrams + Semantic Headword	65.8%
B. Unigrams + First Token	67.4%
B. Unigrams + Binary Length	67.4%
B. Unigrams + Ner-Increment	66.8%
B. Unigrams + Ner-Replace	66.6%
B. Unigrams + Number of Chars	67.4%
B. Unigrams + Number of Tokens	67.6%
B. Unigrams + Phrase	67.2%
B. Unigrams + POS tags	67.8%
B. Unigrams + Pseudo Headword	64%

Accuracy for the Regex Method	
Features	Just Method
Ner-Replace + Binary Word Shape	83.8%
Ner-Replace + Word Shape	83.2%
Ner-Replace + Binary Verb	83.6%
Ner-Replace + Semantic Headword	84.4%
Ner-Replace + First Token	**85.6%**
Ner-Replace + Binary Length	83.2%
Ner-Replace + Ner-Increment	83.8%
Ner-Replace + Number of Chars	83.4%
Ner-Replace + Number of Tokens	83.2%
Ner-Replace + Phrase	84%
Ner-Replace + POS tags	85.4%
Ner-Replace + Pseudo Headword	82.4%
Unigrams + Binary Word Shape	83%
Unigrams + Word Shape	84.2%
Unigrams + Binary Verb	84.4%
Unigrams + Semantic Headword	84%
Unigrams + First Token	**85.2%**
Unigrams + Binary Length	82.6%
Unigrams + Ner-Increment	83.2%
Unigrams + Ner-Replace	83.6%
Unigrams + Number of Chars	84.4%
Unigrams + Number of Tokens	83.6%
Unigrams + Phrase	84.2%
Unigrams + POS tags	85%
Unigrams + Pseudo Headword	81.8%
B. Unigrams + Binary Word Shape	82.8%
B. Unigrams + Word Shape	83.4%
B. Unigrams + Binary Verb	**85.2%**
B. Unigrams + Semantic Headword	83.6%
B. Unigrams + First Token	85%
B. Unigrams + Binary Length	84%
B. Unigrams + Ner-Increment	84.2%
B. Unigrams + Ner-Replace	84.6%
B. Unigrams + Number of Chars	84.4%
B. Unigrams + Number of Tokens	83.6%
B. Unigrams + Phrase	83.8%
B. Unigrams + POS tags	85%
B. Unigrams + Pseudo Headword	81.8%

C Query Ensembles's Base Classifiers

This appendix depicts for both coarse and fine-grained classification, the feature combination of the six base classifiers for each type of classifier considered - SVM-Linear, SVM-RBF and Naive Bayes - as well as their accuracy for the test dataset mentioned in Section 4.3.1.

C.1 Coarse-Classification for Just Method

SVM-Linear	
Features	Accuracy
Unigrams + Binary Word Shape + First Token	68.2%
B. Unigrams + Binary Word Shape + First Token	68.6%
Unigrams + Wordshape + First Token	68%
Unigrams + Number of Chars + First Token	66.6%
Unigrams + Binary Word Shape + Phrase	66.2%
B. Unigrams + Binary Word Shape + Phrase	65.6%

SVM-RBF	
Features	Accuracy
B. Unigrams + Ner-Replace + Semantic Headword	58.4%
Unigrams + Ner-Replace + Semantic Headword	58.2%
B. Unigrams + Ner-Replace + Phrase	57.8%
Unigrams + Binary Verb + Pseudo Headword	57.6%
B. Unigrams + Ner-Replace + Number of Tokens	55.6%
B. Unigrams + Ner-Replace + POS tags	55.2%

Naive Bayes	
Features	Accuracy
Ner-Replace + POS tags + First Token	63.6%
Ner-Replace + Binary Verb + POS tags	63.2%
Ner-Replace + Phrase	62.8%
Ner-Replace + Binary Verb + Phrase	62.2%
Ner-Replace + Phrase + First Token	62.2%
Ner-Replace + Binary Verb	60.2%

C.2 Coarse-Classification for Regex Method

SVM-Linear	
Features	Accuracy
Unigrams + POS tags + First Token	75.8%
Ner-Replace + First Token + Phrase	75.6%
Ner-Replace + First Token	75%
Ner-Replace + POS tags	74.4%
B. Unigrams + POS tags + First Token	75%
Unigrams + POS tags + Binary Verb	74.8%

SVM-RBF	
Features	Accuracy
B. Unigrams + Ner-Replace + Semantic Headword	68%
Unigrams + Ner-Replace + Semantic Headword	67.8%
Unigrams + Ner-Replace + Word Shape	65.8%
B. Unigrams + Ner-Replace + Word Shape	65.8%
Ner-Replace + Semantic Headword + Binary Verb	65.4%
Ner-Replace + Semantic Headword	65.4%

Naive Bayes	
Features	Accuracy
Ner-Replace + POS tags + First Token	69.4%
Ner-Replace + Phrase	69.2%
Ner-Replace + POS tags	68.4%
Ner-Replace + Phrase + First Token	67.6%
Ner-Replace + Phrase + Binary Verb	67%
Ner-Replace + Binary Verb	66.8%

C.3 Fine-Classification for Just Method

SVM-Linear	
Features	Accuracy
B. Unigrams + Word Shape + Phrase	68%
Ner-Replace + Binary Word Shape	60%
Ner-Replace + Word Shape	59.2%
Unigrams + Binary Word Shape + Number of Tokens	59%
Unigrams + Word Shape + Binary Length	57.2%
Ner-Replace + POS tags	56.8%

SVM-RBF	
Features	Accuracy
B. Unigrams + Ner-Replace + Semantic Headword	40.6%
Unigrams + Ner-Replace + Semantic Headword	40.4%
Unigrams + Ner-Replace + Binary Verb	38.8%
B. Unigrams + Binary Verb + First Token	38.8%
Unigrams + Ner-Replace	38.4%
Ner-Replace + Binary Verb + First Token	38.2%

Naive Bayes	
Features	Accuracy
Ner-Replace + POS tags + First Token	54%
Ner-Replace + POS tags + Binary Verb	49.2%
Ner-Replace + POS tags	48.8%
Ner-Replace + POS tags + Ner-Increment	47.8%
First Token + POS tags + Binary Verb	43.2%
First Token + POS tags	38.4%

C.4 Fine-Classification for Regex Method

SVM-Linear	
Features	Accuracy
Ner-Replace	70%
Unigrams + Binary Word Shape + Ner-Replace	69.8%
Binary Word Shape + Ner-Replace	69.6%
Unigrams + Ner-Replace	69.4%
Unigrams + Ner-Replace + Binary Verb	69.2%
B. Unigrams + Binary Word Shape	68.6%

SVM-RBF	
Features	Accuracy
Unigrams + Ner-Replace + Semantic Headword	53%
B. Unigrams + Ner-Replace + First Token	52%
Unigrams + Ner-Replace + First Token	52%
Unigrams + Ner-Replace	51.2%
B. Unigrams + Ner-Replace	51.2%
Ner-Replace + Semantic Headword + Binary Word Shape	47.2%

Naive Bayes	
Features	Accuracy
Ner-Replace + POS tags + First Token	57.8%
Ner-Replace + Binary Verb + Pseudo Headword	56.2%
Ner-Replace + Binary Verb	55.8%
Ner-Replace + Binary Verb + Phrase	54.6%
Ner-Replace + POS tags	54%
Ner-Replace + POS tags + Binary Length	52.8%

D Query Reformulation Patterns

This appendix depicts some of the patterns that are part of the Just.Ask Query Reformulation approached described in Section 5.2.1. More specifically, the Open-Ephyra Patterns/Just.Ask High Level Patterns are shown as well as some examples of Just.Ask Category Level Patterns and the respective queries generated by those patterns. Only *standard* queries are presented since *phrase* queries are identical to standard queries, except that they are surrounded by quotation marks.

D.1 Open-Ephyra Patterns/Just.Ask High Level Patterns

```
How (is|are|was|were) (.*)
When (is|are|was|were) (.*)
Where (is|are|was|were) (.*)
Who (is|are|was|were) (.*)
Why (is|are|was|were) (.*)
(?:What|Which) (is|are|was|were) (.*)
How (.*)
When (.*)
Where (.*)
Who (.*)
Why (.*)
(?:What|Which) (.*)
Name (.*)
```

D.2 Just.Ask Category Level Patterns Examples

D.2.0.1 LOCATION_CITY

Pattern: `What ('s|is) the capital of (.*)`
Question: What is the capital of Somalia ?
Queries: the capital of Somalia is

Somalia capital is
is the capital of Somalia

D.2.0.2 ENTITY_SUBSTANCE

Pattern: `What ('s|is|are|was|were) (an|a|the) (.*) made (out|of|from)`
Question: What is the Chicken Boy sculpture made of ?
Queries: the Chicken Boy sculpture is made with
the Chicken Boy sculpture is made from
the Chicken Boy sculpture is made of
the are the components of the Chicken Boy sculpture
the are the constituents of the Chicken Boy sculpture

D.2.0.3 NUMERIC_DATE

Pattern: `When (did|does|do|is|are|was|were|will) (.*)`
Question: When did Innsbruck host the Winter Olympics ?
Queries: in Innsbruck hosted the Winter Olympics
Innsbruck hosted the Winter Olympics in
is the date when Innsbruck hosted the Winter Olympics
is the year when Innsbruck hosted the Winter Olympics

D.2.0.4 NUMERIC_PERIOD

Pattern: `How long (did|was|were|is|are|has|would) (.*)`
Question: How long did Mark Kauffman work for the Life magazine ?
Queries: Mark Kauffman worked for the Life magazine during centuries
Mark Kauffman worked for the Life magazine during decades
Mark Kauffman worked for the Life magazine during years
Mark Kauffman worked for the Life magazine during moths
Mark Kauffman worked for the Life magazine during weeks
Mark Kauffman worked for the Life magazine during days
Mark Kauffman worked for the Life magazine for centuries
Mark Kauffman worked for the Life magazine for decades
Mark Kauffman worked for the Life magazine for years
Mark Kauffman worked for the Life magazine for moths
Mark Kauffman worked for the Life magazine for weeks

Mark Kauffman worked for the Life magazine for days

D.2.0.5 ABBREVIATION_EXPANSION

Pattern: `What (does|do) the (acronym|abbreviation) (.*) (stands|stand) for`
Question: What does the abbreviation GIA stand for ?
Queries: GIA is the abbreviation of
GIA is the abbreviation for
GIA is an abbreviation of
GIA is an abbreviation for
GIA is an acronym of
GIA is an acronym for
GIA stands for
GIA stand for
GIA means

D.2.0.6 NUMERIC_COUNT

Pattern: `How many (.*)`
Question: How many former slaves settled in Liberia in the 19th century ?
Queries: is the number of former slaves settled in Liberia in the 19th century
number of former slaves settled in Liberia in the 19th century round
millions of former slaves settled in Liberia in the 19th century
thousands of former slaves settled in Liberia in the 19th century
hundreds of former slaves settled in Liberia in the 19th century

D.2.0.7 DESCRIPTION_MANNER

Pattern: `How (did|does|do) (.*)`
Question: How did Vitas Gerulaitis die ?
Queries: Vitas Gerulaitis died of
Vitas Gerulaitis died from
Vitas Gerulaitis died with

D.2.0.8 HUMAN_INDIVIDUAL

Pattern: `Who (\\w+)ed (.*)`

Question: Who founded the FHP health care services ?
Queries: the FHP health care services were developed by
the FHP health care services were invented by
the FHP health care services were founded by
was the creator of the FHP health care services
was the developer of the FHP health care services
was the inventor of the FHP health care services
were the developers of the FHP health care services
were the inventors of the FHP health care services

Pattern: `Who (wrote|created|produced) the (play|book|song|poem|history|lyrics|script|argument|letter)* (.*)`
Question: Who wrote ”Gulag Archipelago” ?
Queries: writer created the ”Gulag Archipelago”
writer wrote the ”Gulag Archipelago”
”Gulag Archipelago” was created by
”Gulag Archipelago” was written by

D.2.0.9 HUMAN_TITLE

Pattern: `What (was|is) (.*) title (.*)`
Question: What is the title of the third album of the Offspring ?
Queries: the third album of the Offspring possesses the title
the third album of the Offspring title is
the third album of the Offspring has the title
is the title of the third album of the Offspring

D.2.0.10 HUMAN_GROUP

Pattern:
`For which (organization|company|corporation) (does|do) (.*) (works|work)`
Question: For which company does Marty McKewon work ?
Queries: Marty McKewon is an employee of
Marty McKewon is an employee of the company
Marty McKewon is an employee of the corporation
is the employer of Marty McKewon
Marty McKewon works for

Marty McKewon works for corporation
Marty McKewon works for company

D.2.0.11 NUMERIC_MONEY

Pattern: `What (is|are|was|was|were) (.*)`
Question: What is the highest fine that can be levied on a conspirator ?
Queries: the highest fine that can is be levied on a conspirator is
the highest fine that can is be levied on a conspirator is $
the highest fine that can is be levied on a conspirator is dollars
the highest fine that can is be levied on a conspirator is €
the highest fine that can is be levied on a conspirator is euros

D.2.0.12 LOCATION_OTHER

Pattern: `Where (was|were) (.*) born`
Question: Where was Guillermo Ortiz born ?
Queries: Guillermo Ortiz birthplace is
Guillermo Ortiz was born in
Guillermo Ortiz was born on
Guillermo Ortiz was born at

D.2.0.13 NUMERIC_DISTANCE

Pattern: `How (deep|tall|wide|ample|broad|long|extensive|prolonged|far) (is|are|was|were) (.*)`
Question: How long is the coastline of Santa Monica Bay ?
Queries: the coastline of Santa Monica Bay is km long
the coastline of Santa Monica Bay is m long
the coastline of Santa Monica Bay is ft long
the coastline of Santa Monica Bay is kilometers long
the coastline of Santa Monica Bay is meters long
the coastline of Santa Monica Bay is feet long
the coastline of Santa Monica Bay is miles long

D.2.0.14 LOCATION_OTHER

Pattern: `Where did (.*) die ?`

Question: Where did Irving Mitchell Felt die ?
Queries: Irving Mitchell Felt death place is
Irving Mitchell Felt death location is
Irving Mitchell Felt died in
Irving Mitchell Felt died at
Irving Mitchell Felt died on

Pattern: `(What|Where) ('s|is|are|was|were) (.*)`
Question: Where is the organization Operation Unity based ?
Queries: the organization Operation Unity based is at
the organization Operation Unity based is on
the organization Operation Unity based is in
the organization Operation Unity based location is
is the location of the organization Operation Unity

E Query Expansion Results

This appendix presents the all results obtained in the evaluation of the individual and conjugated contributions of expanding the question headword/compound headword and the main verbs of a query. The results are separated according to the type of expansion performed and the search engine that was utilized.

E.1 Headword/Compound Headword Expansion Bing Results

Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	140 (76,5%)	0.40	0.52
40%	140 (76,5%)	0.40	0.52
60%	**140 (76,5%)**	**0.40**	**0.53**
80%	140 (76,5%)	0.40	0.52
100%	140 (76,5%)	0.40	0.52
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	129 (70,4%)	0.32	0.45
40%	129 (70,4%)	0.32	0.46
60%	129 (70,4%)	0.33	0.47
80%	**129 (70,4%)**	**0.35**	**0.49**
100%	**129 (70,4%)**	**0.35**	**0.49**

E.2 Headword/Compound Headword Expansion Lucene Results

Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	104 (56,8%)	0.19	0.33
40%	105 (57,3%)	0.19	0.34
60%	107 (58,4%)	0.19	0.32
80%	110 (60,1%)	0.19	0.32
100%	**111 (60,6%)**	**0.19**	**0.32**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	99 (54%)	0.18	0.34
40%	100 (54,6%)	0.19	0.35
60%	101 (55,1%)	0.19	0.35
80%	104 (56,8%)	0.19	0.34
100%	**106 (57,9%)**	**0.2**	**0.32**

E.3 Main Verbs Expansion Bing Results

Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	143 (78,1%)	0.39	0.5
40%	145 (79,2%)	0.42	0.52
60%	148 (80,8%)	0.42	0.52
80%	149 (81,4%)	0.43	0.53
100%	**150 (81,9%)**	**0.44**	**0.53**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	133 (72,6%)	0.33	0.45
40%	136 (74,3%)	0.33	0.45
60%	140 (76,5%)	0.35	0.46
80%	140 (76,5%)	0.36	0.47
100%	**140 (76,5%)**	**0.37**	**0.48**

E.4 Main Verbs Expansion Lucene Results

Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	106 (57,9%)	0.18	0.31
40%	107 (58,4%)	0.18	0.31
60%	107 (58,4%)	0.18	0.31
80%	108 (59%)	0.18	0.31
100%	**110 (60,1%)**	**0.19**	**0.31**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	96 (52,5%)	0.16	0.3
40%	97 (53%)	0.16	0.3
60%	97 (53%)	0.16	0.3
80%	100 (54,6%)	0.16	0.3
100%	**102 (55,7%)**	**0.17**	**0.3**

E.5 Headword/Compound Headword and Main Verbs Expansion Bing Results

Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	142 (77,5%)	0.40	0.52
40%	144 (78,6%)	0.43	0.54
60%	146 (79,7%)	0.44	0.54
80%	147 (80,3%)	0.45	0.55
100%	**147 (80,3%)**	**0.45**	**0.56**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	131 (71,5%)	0.32	0.44
40%	134 (73,2%)	0.33	0.45
60%	137 (74,8%)	0.35	0.47
80%	138 (75,4%)	0.37	0.48
100%	**138 (75,4%)**	**0.38**	**0.5**

E.6 Headword/Compound Headword and Main Verbs Expansion Bing Results

Query Expansion Formulator with Keyword Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	106 (57,9%)	0.18	0.31
40%	107(58,4%)	0.19	0.32
60%	109 (59,5%)	0.18	0.31
80%	113 (61,7%)	0.19	0.3
100%	**115 (62,8%)**	**0.19**	**0.3**
Query Expansion Formulator with Regex Query Formulator - 50 passages			
Number of Expansion Terms	#Question$_{1+PosPassage}$	MRR$_{ALLQ}$	MRR$_{QPOSPASSAGES}$
20%	97 (53%)	0.17	0.32
40%	98 (53,5%)	0.17	0.32
60%	99 (54%)	0.18	0.33
80%	105 (57,3%)	0.18	0.32
100%	**108 (59%)**	**0.19**	**0.32**