QUESTION ANALYSIS AND INFORMATION RETRIEVAL FOR A TURKISH QUESTION ANSWERING SYSTEM: HAZIRCEVAP

by

Caner Derici

B.S., Computer Science, İstanbul Bilgi University, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2014

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Tunga Güngör for his support during the final stages of my graduate study at Boğaziçi University. Besides providing a wonderful working environment and lovely a group of people to work with, he has patiently devoted his time and energy to me in all the stages of this study.

I would like to thank Arzucan Özgür for very helpful discussions, for reading and commenting on early versions of this study. Her ideas and suggestions about both the models and the academic writing have played a critical role in this thesis.

Also, I thank Günizi Kartal for accepting to be in my thesis committee, and devoting his time to read the dissertation.

Additionally, I am grateful to Kerem Çelik for long Skype sessions, Günizi Kartal for her valuable comments on academic language as well as the data and question class preparation for which I am most grateful as well to the efforts of Yiğit Aydın and Ekrem Kutbay.

We acknowledge that this study is fully supported by Tubitak under the project identified as 113E036.

I also would like to thank my family and friends for their invaluable support in all of my educational journey.

Finally, I am most grateful to my sister, Özlem Derici, and my girlfriend, Dicle Yalçın. If it were not for my sister, I could never hope to chase my dreams, and if not for Dicle, I could not be who I am today. I thank both of you to shape my life and my mind the way that it is now.

ABSTRACT

QUESTION ANALYSIS AND INFORMATION RETRIEVAL FOR A TURKISH QUESTION ANSWERING SYSTEM: HAZIRCEVAP

This study describes and evaluates the techniques we developed for the question analysis and information retrieval (IR) module of a closed-domain Turkish factoid Question Answering (QA) system that is intended for high-school students to support their education. Question analysis, which involves analyzing the questions to extract the necessary information for determining what is being asked and how to approach answering it, is one of the most crucial components of a QA system. Therefore, we propose novel methods for two major problems in question analysis, namely focus extraction and question classification, based on integrating a rule-based and a Hidden Markov Model (HMM) based sequence classification approach, both of which make use of the dependency relations among the words in the question. We also investigate the IR module, which is another critical aspect of a QA system, and introduce the IR module to efficiently gather the relevant information to a given question, with which the answer will be determined. IR module searches for the relevant documents and passages through the combined use of search engines Indri and Apache Lucene. Solution to these problems constitute the framework, on top of which a whole QA system can easily be built with only an addition of an answering module. Comparisons of all solutions with baseline models are provided. This study also offers a manually collected and annotated gold standard data set for further research in this area.

ÖZET

TÜRKÇE SORU CEVAPLAMA SİSTEMİ İÇİN SORU ANALİZİ VE BİLGİ ÇIKARIMI: HAZIRCEVAP

Bu çalışmada lise öğrencilerinin eğitimlerine yardımcı olması için geliştirilen kapalı-alan Türkçe tek cevaplı Soru Cevaplama (SC) sisteminin inşasında tasarlanan soru analizi ve bilgi çıkarımı (BÇ) modülleri için geliştirilmiş teknikler anlatılmakta ve değerlendirilmektedir. Verilen bir soruda tam olarak neyin sorulduğu ve cevaplamanın ne şekilde yapılması gerektiğini belirlemek için sorudan gerekli bilgileri çıkartan soru analizi, bir soru cevaplama sisteminin en önemli parçalarından biridir. Bu nedenle bu çalışmada soru analizindeki en önemli iki problem olan odak çıkarımı ve soru sınıflandırılması problemlerine, kural tabanlı ve Saklı Markov Modeli (SMM) tabanlı modellerin sentezinden oluşan ve sorudaki kelimeler arasındaki bağlılık ilişkilerini kullanan çözümler sunulmuştur. Ek olarak bir SC sisteminin bir başka önemli modülü olak BÇ modülü de incelenmiş, ve içerisinde verilen sorunun cevabının aranacağı ilgili bilgileri kümesinin verimli bir şekilde çıkartılması için de teknikler önerilmiştir. BÇ modülü, soru ile ilgili döküman ve pasajları Indri ve Apache Lucene arama motorlarını kullanarak bulmaya çalışmaktadır. Sunulan çözümler, üzerine sadece cevap modülünün eklenmesiyle tam bir SC sisteminin oluşturulabileceği bir altyapı oluşturmaktadır. Önerilen tüm çözümlerin karşılaştırmalı deneyleri, baz modelleri ile birlikte sunulmuştur. Bu çalışmada aynı zamanda, elle toplanıp işaretlenmiş Türkçe standard veri kümesi, bu alanda daha sonraki araştırmalarda kullanılmak üzere genel kullanıma açılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST ACRONYMS / ABBREVIATIONS

HMM	Hidden Markov Model
IR	Information Retrieval
LAT	Lexical answer type
NLP	Natural Language Processing
POS	Part-of-speech
QA	Question Answering
SVM	Support Vector Machine

1. INTRODUCTION

Propensity to find new ways to process and draw meaning out of massively growing information is increasing in global scale with tremendous efforts of researchers from lots of different fields. In the field of natural language processing (NLP) Question Answering (QA) problem aims to produce automatically generated answers for questions stated in natural languages. The drastic improvements in NLP, particularly in Information Retrieval (IR) techniques in the past decade have led to the development of prominent QA systems, some of which are available for public use, such as AnswerMachine[1] and WolframAlpha[2] . It has even been possible to develop a QA system that can compete on a TV show against human opponents [1]. These systems try to process previously collected information (usually world-wide-web) and helps users find meaningful results for their information search. QA systems' purpose is to produce specific answers, instead of finding related answers or documents (or web pages) considered to contain the answer, as most of the general purpose search engines do. Some studies try to go even further to answer open-ended questions like "what is the effect of climate change to flora in South America?", or even further to questions like "what would be the effects of a war between Israel and Palestine to the world economy?". While it seems impossible in reality for a computer to answer such questions, as they are considerably difficult to answer even for humans, QA systems have the advantage over humans to process massive information, produce a meaningful compilation and analyze them within minutes. QA researchers work in this regard, try to find novel analysis and answer extraction techniques to build powerful QA systems.

1.1. Question Answering Problem

Building a fully capable question answering system, however, has difficulties mostly originating from the fact that there are numerous challenging sub-problems that need to be solved, such as question analysis (involving pre-processing and clas-

[1] http://theanswermachine.tripod.com/
[2] http://www.wolframalpha.com/

sifying the questions), information retrieval, cross linguality (to accept questions and produce answers in different languages and to be able to use resources from any language) and answer generation (involving the extraction and formulation of the answer), along with some lower level sub-tasks such as paraphrasing, common sense implication or reference resolution. In addition, the architecture of a QA system, as well as the techniques employed usually depend on factors such as the question domain and the question language. Typically, a question answering system has a knowledge base to consult when generating an answer, along with a pipeline of individual modules with increasing complexities, where each module tries to solve one of the intrinsic problems stated above, while producing meta information to be used by subsequent modules of the system. In the end, an answer is compiled from the resources and formulated, as a result of the combined effort of each inner module in the system. Many researchers tried to tackle the individual problems involved in such systems separately over the years. While some of them are considered to be solved, majority of the problems are still open to further research [2, 3].

This study introduces a closed-domain factoid question answering system designed for Turkish, namely HazırCevap (literally means PromptAnswered, an idiom in Turkish describing an entity being quick and accurate in answering any kind of question). The system is developed for high-school students to enable them to query in their natural language any question chosen from their courses of study. Note that, there is virtually no upper bound in the number of possible query frequency (number of distinct questions), as HazırCevap is anticipated to be used by virtually all high schools throughout the nation. Therefore in order for HazırCevap to be practically usable, it is of utter significance that besides the accuracy, the overall system architecture should be carefully designed, which have led us to believe that each module of the system should be comprehensively analyzed and evaluated in separate studies. In this study, we present the development and evaluation of two modules, namely question analysis and information retrieval modules in the pipeline of HazırCevap, to be used to analyze questions in our prototype domain of Geography and produce relevant documents and passages in which the answer will be looked for. The primary concern in question analysis is to extract useful information from a given question, to be used in subsequent

modules to finally generate a correct response. In particular, the information that indicates a certain type or a property for the entity that is being asked for, along with a classification of the question into pre-determined classes from the domain helps to reduce significantly the size of the work space of the further stages of the system such as information retrieval or candidate answer generation. For instance, in the following example, the information indicating that we are searching for a name of a plain, which we refer to as the focus of the question, as well as the question classification as ENTITY.PLAIN helps us to easily navigate around these concepts through the knowledge base.

"Türkiye'nin en büyük ovasının adı nedir?"

"What is the name of the largest plain in Turkey?"

For focus extraction, we developed a rule-based model, along with a Hidden Markov Model (HMM) based statistical model. We investigate the accuracy of the combination of these two in focus extraction. Additionally, for question classification, we show that a rule-based model is more successful in finding coarse classes than a tf-idf based bag-of-words baseline model that utilizes the frequencies of the words in a question. Developing such a question analysis module, let alone a QA system for Turkish is especially challenging because it is an agglutinative language with a morphologically rich and derivational structure. For this reason, we pre-process the questions by performing morphological analysis and disambiguation, as well as dependency parsing using the Turkish NLP Pipeline [4, 5, 6]. Morphological analysis and disambiguation produces the root forms of the words and their part-of-speech (POS) tags. Dependency parsing produces the dependency relations among the words in a given sentence. The tags that are used by the dependency parser are defined in the Turkish Dependency TreeBank, which includes tags such as SUBJECT, OBJECT, SENTENCE, MODIFIER, CLASSIFIER, POSSESOR, and etc. [5, 7].

In addition to question analysis, we also investigate the methods to retrieve rel-

evant information to a given question. Designed IR module tries to find relevant document and passages through the use of programmable general purpose search engines such as Indri and Apache Lucene. Automatically generated and expanded queries for a given question are used by these search engines to produce relevant passage and documents.

We propose a novel approach for question classification and focus detection, based on integrating a rule-based method with an HMM-based sequence classification method, in addition to two different (i.e. unstructured and structured) IR approaches for a closed-domain factoid QA system. Additionally, we contribute the first manually collected and annotated gold standard question analysis data set for Turkish. The implementation codes and the gold standard Turkish question data are publicly available for reproducibility and further research[3] .

1.2. Related Work

A fundamental task in a QA system is determining the type of the answer, and its properties and possible constraints. Given a query stated in a natural language, a QA system often extracts some immediate information such as the question class (e.g. what, who, when, etc.) based on the pre-determined answer types [8]. Recent state-of-the-art techniques for question classification often involve statistical approaches [9, 10]. Additionally, some QA systems are in general more semantics oriented, and construct a knowledge base directly from raw question texts [11]. However, these systems determine only the type of a given question. They do not further determine, for example what type of entity is being asked, which would narrow down the search space significantly.

One approach in simulating a question analysis is to use general purpose search engines. One of the earliest studies that employs such a strategy, is an open-domain QA system, AnswerBus [12]. AnswerBus employs a bag-of-words strategy, where search engines are scored based on the number of hits they returned for each word. The total score of a search engine for a particular question is the sum of the hits returned

[3] https://github.com/cderici/hazircevap

for each word in the question. Based on their total scores, the best search engine is determined as the most appropriate knowledge source for answering the question. However, AnswerBus does not use any semantic information, nor does it extract any information to build a more competent answering strategy. Another deficiency in AnswerBus seems to be the fact that it doesn't produce a direct answer. Instead, it produces web pages that are most likely to contain the answer. Finding the actual answer has been left to the user. In this respect, it is more closer to being a search engine rather than a QA system. Another early QA system, unlike AnswerBus, performs question analysis as well as answer synthesis is named AskMSR [13]. It performs question reformulation to produce different variations of the given question in order to increase the probability of finding an answer match. Then it feeds these questions to a search engine that queries the database for related text passages, that are in turn used for learning unigram, bigram and trigrams. In the end, AskMSR synthesizes an answer based on these previously learned weighted n-grams. Being a fully statistical system, it shares the deficiency of not using any semantic information, in addition to the inefficiency of performing highly intensive computations to learn n-grams on the fly.

There are two general choices regarding the questions in building a QA system. First concern is that questions can be factoid (i.e. with only one possible answer) or open-ended (which involves humane interpretation to some extend). Factoid questions are generally considered to be easier than open-ended questions, since rather than extracting the links between facts and concepts and producing a compilation, searching a single bit of information as an answer is just a matter of computational power. Second concern is that questions may be drawn from a single particular domain in a closed-domain system, or the system can be an open-domain system and the questions can be about any domain. Studies conducted in the last years concentrated mostly on open-domain QA systems, and try to build a system that isn't constrained by a particular domain. These systems try to find the answer best matching the question by mostly machine learning statistical methods such as language models. For example, a system designed to work on Turkish finds the best matching answer using text mining [14]. Another system for Turkish learns the most frequent patterns occurring in both

questions and answers [15]. It then matches the question-patterns and answer-patterns and returns the sentence or passage that mostly contains the similar patterns as the answer.

Another approach in open-domain systems is to entirely rely on a logical framework. These systems convert a given question into a logical formula and try to find the answer through logical manipulations. A logical framework for such a system (in Turkish) is developed by Say, namely TOY [16]. The platform has a morphological analyzer, lexicon and a knowledge base, making possible that a fully functional QA system is to be build upon. As a result of its design characteristics, TOY can perform complex logical transformations in order to infer the answer that the user is looking for.

The intermediate representation that the given question is transformed doesn't have to be a logical formula. Systems that are more general define their own intermediate representations and transform the given questions into these representations. Having a special intermediate representation, these systems can perform their own analysis specialized to their domain of use. One of the most advanced system that uses its own representation is START, which is developed at MIT University [11]. START parses and annotates a given question in order to dissect it into useful and meaningful parts. Having templates such as <Subject - Relation - Object>, these parts are used to look for similar parts in the knowledge base by previously defined relation transformations.

Unlike the systems try to go directly for an answer from the question, START and TOY are examples of systems that perform various analyses to extract information that make possible detailed and smart search strategies. In the last years, question classification and reformulation were the most popular analysis and preparation techniques. More advanced systems generally rely on machine learning based statistical methods to perform information extraction and logical answer synthesis. One of the most sophisticated QA system in this regard, is IBM Watson [17].

Inspired by its significant success, our system adapts its strategies for question analysis and information retrieval among the ones that are employed in one of the most powerful QA systems, IBM's Watson [18]. For analyzing a given question (i.e. clue), Watson extracts firstly a part of the clue that is a reference to the answer (focus); second, it extracts the terms that denote the type of the entity asked (lexical answer type, LAT); third, the class of the clue (QClass); and finally some additional elements of the clue (QSection) should it need special handling. Lally et al. [18] extensively evaluate the significance of distilling such information on end-to-end performance (i.e. produce correct answers). To extract these information, Watson mostly uses regular expression based rules combined with statistical classifiers to assess the learned reliability of the rules. Note that, the sole purpose of Watson is to win the Jeopardy! game, a well-known television quiz show where the quiz questions are presented as free formatted "clues", rather than complete question statements, rendering Watson's analysis methods specific to the Jeopardy! game. In a closed-domain QA system, on the other hand, it is sufficient to extract only LAT and QClass in order to analyze a complete question, since in a complete question sentence, what Watson refers to as the focus is often the question word (e.g. "What" in the example in Section 1). Therefore, the real focus of a question, what we refer to as the focus is closest in reality to what Watson refers to as LAT.

For information retrieval, Watson relies on searching through pre-processed unstructured resources using search engines such as Indri to retrieve relevant passages and documents. It also employs a statistical structural look up technique, named Prismatic, which works on aggregate statistics collected by pre-processing all the textual content [19]. Watson mainly uses the recent snapshot of Wikipedia (more than four million documents with over 13 GB of textual content) as a base corpus, in addition to Wiktionary, Wikiquote, Bible and various books from Project Gutenberg. Additionally Watson employs a statistical scoring algorithm to expand its sources with Web documents. With these source acquisitions and expansions in total, Watson searches through over eight million documents sizing around 60 GB textual data [20].

2. DEEPQA AND HAZIRCEVAP ARCHITECTURE

In this chapter, we introduce the overall architecture and design decisions behind HazırCevap. In order for a complete picture of the system, it is necessary to understand the basic principles of DeepQA, a general question answering architecture upon which our system is being built, in addition to the implementation of Watson on DeepQA.

DeepQA is fully parallel, probabilistic evidence based computation architecture. It employs various kinds of analysis techniques placed in parallel to each other in order to work on different parts of the question answering process. Being a completely probabilistic architecture, each analysis module produce also a confidence score along with its results, indicating how strong the analyzer trusts its own results. This provides the necessary dexterity for the system to eliminate bad choices (i.e. choices with low confidence scores) early, especially before they are used to search for relevant documents and candidate answers, which are considerably costly operations. Because most of the parts of the system works in parallel to each other, it can handle heavy-duty computations with much less effort, thereby increasing its efficiency and thus practical ease of use [17].

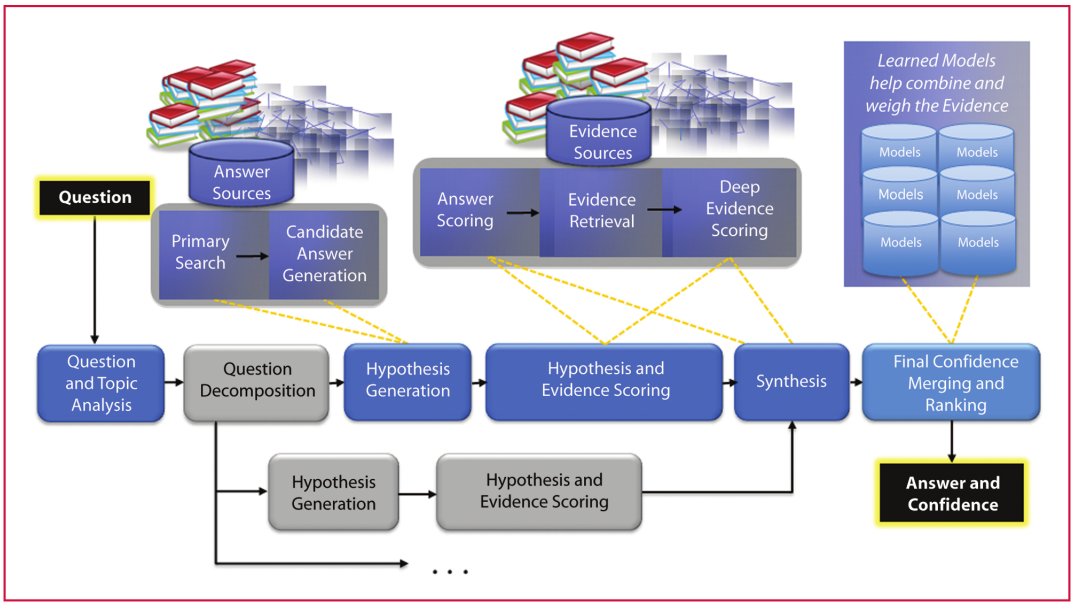


Figure 2.1. IBM Watson with the DeepQA Architecture [1].

Given a question, IBM Watson firstly feeds it to the question analysis module. This module extracts some immediate information such as lexical answer type (LAT) and question class, which are later used by subsequent modules [18]. Then the decomposition module breaks the question into its smaller parts, and Watson re-compiles these parts in different ways in question reformulation module, thereby creating different variations of the initial question [21]. Up to this point, the system works in a fairly deterministic way (i.e. the use of statistical methods is fairly slim). In hypothesis (set of candidate answers) generation stage, Watson starts to search through previously compiled information base by previously trained machine learning models with an additional help of customized search engines [20, 19]. Collected candidate answers for each variation of the original question then filtered by type coercion module using the criteria information extracted by question analysis module. The candidates having incompatible types are eliminated [22]. Then the evidence collector module, having multiple trained machine learning models, try to gather all kinds of evidences for each candidate answer. These evidences are then used to compute each candidate's confidence scores [23]. Final set of candidate answers and confidence scores are then fed to the final ranking and synthesizing module. This module again uses multiple machine learning models to compute the confidence scores from the collected evidences [24]. The candidate answer that has the highest confidence score is chosen to be the answer, and reported as the final answer along with its evidences and confidence score. The whole answering process is depicted in Figure 2.1.

2.1. HazırCevap

In this section we present the overall architecture of HazırCevap, as well as where the question analysis module fits in this architecture. The inner mechanisms of the question analysis module are also described.

Recall that HazırCevap is designed to be used by high-school students to help with their education. In practice, students will as factoid questions (i.e. questions with a single objective answer) in Turkish, and the system will generate a single answer, along with the context that is compiled from the passages that the answer has been

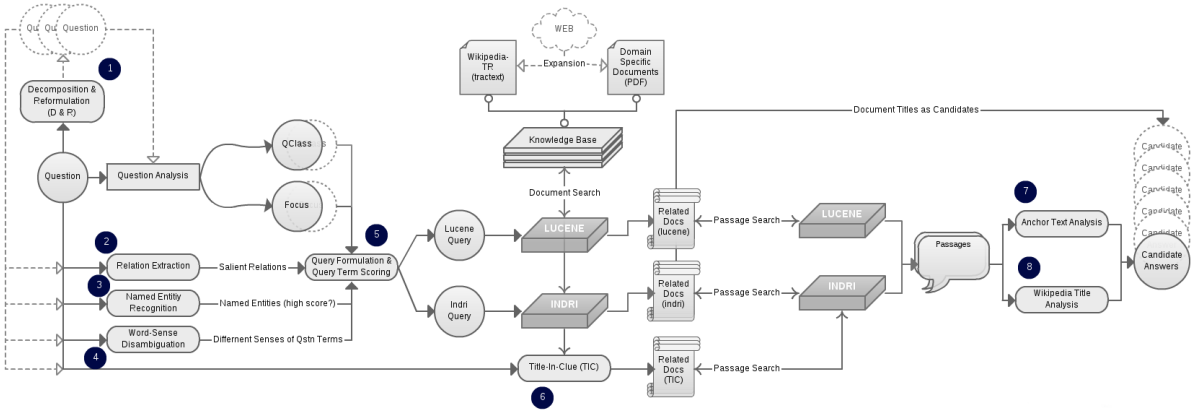


Figure 2.2. HazırCevap System Architecture.

found.

One of the most crucial design decisions of such a system is the domain. Typically, the open-domain and the closed-domain QA systems have distinct characteristics for each sub-problem such as information retrieval or candidate answer generation. This distinction affects directly the architecture of the system. For example, the domain specific information may prove to be useful for closed-domain systems, such as GETARNUS [25], while in contrast, open-domain systems should have strong information retrieval capabilities to handle virtually all possible domains, along with a competent computational architecture to be practically useful. Additionally, the knowledge base on which the system searches the answer may be limited by both the domain and the user profile in closed-domain systems (e.g. if the system is designed to assist pilots while piloting an airplane, the knowledge base may exclude the conference papers on aerodynamics). Conversely, there is essentially no limit on the size of the knowledge base for an open-domain system, such as in MIT START [11]. Therefore the choice of domain setting is a function of the system's use as much as it is a design choice. In this respect, HazırCevap is naturally closed-domain, as it is planned to be used by high-school students in their courses of study defined and constrained by the curriculum.

The overall architecture of the HazırCevap system is designed in concordance with the DeepQA technology. The anticipated architecture of HazırCevap has a design

similar to the DeepQA technology, as illustrated in Figure 2.2. After the question analysis, the extracted focus is used in the Semantic Analysis module to fetch semantically relevant knowledge units that are pruned and refined by the QClass. We refrain from referring to these units as "documents", as we ideally do not limit the format in which the knowledge is represented. These relevant knowledge units are then fed to the Candidate Answer Generation module that has multiple different information retrieval algorithms to produce all possible relevant answer units. Then for each candidate answer unit, multiple trained machine learning algorithms collect and compile both syntactic and semantic evidences, by also consulting to the evidence base that is pre-compiled from the knowledge base. All these evidences are used to score the candidate answer units, of which the ones having low scores are pruned. Finally the strong candidates are synthesized into the final answer set, where the most likely answer is fed to the answer generation module along with the other top k answers for providing optionality.

As shown in Figure 2.2, HazırCevap includes multiple modules for each subproblem in the answering process. The anticipated sub-modules and their effects to the overall system are discussed below.

2.1.1. Decomposition & Reformulation

Decomposition and reformulation disintegrate a given question into small chunks of information and reintegrate these parts in different ways (if possible) to produce different variations of the original question. This increases the versatility of the system, thereby increasing its contrast in processing and analyzing the question, as well as filtering the candidate answers. For example consider the following question:

"Türkiye'nin en büyük ovasında bulunan toprak çeşitleri nelerdir?"

"What types of soil can be seen in the largest plain in Turkey?"

If this question is to be answered, there are two bits of information that need to be extracted; the largest plain in Turkey, and the types of soils. To achieve this, the system needs to decompose the question into two and evaluate these questions separately. The first question produced by the decomposition of the question above is:

"Türkiye'nin en büyük ovası hangisidir?"

"Which is the largest plain in Turkey?"

When the system finishes to process this question, and finds the answer (Konya plain), it reformulates the second question produced by the initial decomposition, and obtain:

"Konya ovasında bulunan toprak çeşitleri nelerdir?"

"What types of soil can be found in Konya plain?"

Additionally, each and every question that is given to the system is reformulated to produce different expressions. The system starts itself with each different expression to work on parallel. This way the probability of finding a correct answer is greatly increased. For example the question "Türkiye'nin en büyük ovasının adı nedir?" (*"What is the name of the largest plain in Turkey?"*) can be reformulated in different ways as follows:

- Türkiye'nin en büyük ovasına ne ad verilir? (*What is the largest plain in Turkey called?*)
- Türkiye'nin en büyük ovası hangisidir? (*Which one is the largest plain in Turkey?"*)
- Türkiye'deki ovaların en büyüğü hangisidir? (*Which plain is the largest of all plains in Turkey?"*)

2.1.2. Relation Extraction

Relation extraction distill the implicit semantic relations among the words in the question and helps the query formulation and improves the performance of the search engines trying to bring the relevant documents and passages. For example, "IS-A" relations helps pruning the candidate answers by type coercion, such as *is_a(Konya, CITY)*, as well as increases the weights of the city terms in the query, which will be fed to the search engine.

2.1.3. Named Entity Recognition

Named entity recognition, like relation extraction, helps the search engines to find the related documents and passages. For example, consider the question below:

"Türkiye'de Ağrı Dağı'ndan sonraki en büyük dağ hangisidir?"

"What is the largest mountain after Ağrı Mountain in Turkey?"

The question asks for a mountain. The information indicating there is a mountain mentioned in the question (Ağrı Mountain) increases the performance of the search engines to find better related documents that are more likely to contain the correct answer. Because if the search engine considers the Ağrı Mountain as well, the found documents and passages will likely to contain the relation between Ağrı Mountain and the mountain that is asked. Therefore the probability of finding "largest mountain after" relation between the answer and the Ağrı Mountain is increased, thereby the probability of finding the answer is increased.

2.1.4. Word-Sense Disambiguation

As in all natural language processing applications, in question answering, it is widely known that determining the used sense of a word often becomes critical for the

overall performance of the system. For example, consider the word "yazı" (*its summer*) in the following question:

"Güney yarımkürenin yazı hangi ayda başlar?"

"Which month is the beginning of the summer in the south hemisphere?"

The word "yazı" also means "writing" in Turkish. If the sense of this word cannot be determined by the system, lot of non-related content will be drawn from the knowledge base by the search engine, causing the system to answer incorrectly, which is often considered as worse than idle response (i.e. the incapability of finding a suitable answer).

2.1.5. Query Formulation & Term Scoring

Query Formulation and Term Scoring module is critical in finding the relevant document and passages that are likely to contain the correct answer to a given question. All the relevant information such as the terms in the question, question words, named entities, relations, word-senses, focus and QClass are combined in this module to correctly generate a query for the search engine and weigh the query terms to help the search engine navigate through the vast space of the previously compiled knowledge base. The class of the question (determined in the question analysis module) determines a scoring template, where the focus, question words and all different types of terms are scored accordingly.

2.1.6. Title-in-Clue (TIC) Search

Parallel to the search engines running on the query that is prepared as explained above, this module searches through the titles of the documents in the knowledge base. The documents having titles that are matching with a word or a phrase in the given question are appended to the set of related documents produced by the search engines.

For example, consider the following question:

"Konya Ovası'nda görülen bitki örtüsü hangisidir?"

"Which type of vegetation can be found in Konya plain?"

Among the documents that are found by search engines, the document titled "Konya Ovası" (in Turkish Wikipedia) is also added to the set of related documents.

2.1.7. Anchor Text Analysis

Once the related documents and passages are selected from the knowledge base, the search for candidate answers begin. One of the analysis techniques that is used to find a candidate answer in a related document is Anchor Text Analysis. It looks for the anchors (hyper-links) in the document and analyses the titles of the documents that the anchors point to. The similarity or the relatedness level of a particular title to the given question (or its focus) determines if the title is entitled to be among the candidate answers. The similarity is measured by various criteria, such as type or named entity class. Related titles are added to the set of candidate answers.

2.1.8. Wikipedia Title Analysis

Similar to the anchor text analysis, in Wikipedia title analysis, related documents and passages are searched for Wikipedia titles. Any word or word group that has its own Wikipedia article is checked for similarity as explained above and added to the candidate answers list if it is computed to be similar to the question (or its focus), discarded otherwise.

The final stages of the system involve multiple previously trained machine learning models trying to search for evidences for candidate answers and scoring them accordingly. Final ranking of confidence scores indicate the correct answer among the

candidates with a certain error threshold. The candidate with the highest confidence score is considered to be the correct answer. The correct answer is synthesized with the confidence score and evidences to compile and produce a proper response. The evidences used in the answering are presented with the links to their originated documents, thereby providing a context for the user to further investigate the answer.

2.2. Question Analysis Module

The purpose of the question analysis module in HazırCevap is to extract some useful information (i.e. focus and QClass) from the given question, in order for HazırCevap to determine what the question is really asking for and how to approach producing an answer.

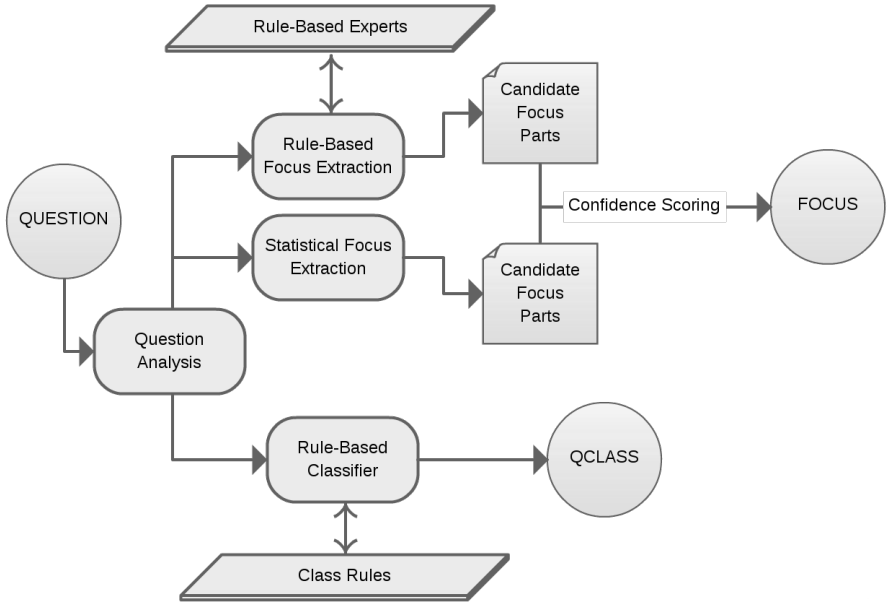


Figure 2.3. Question Analysis Module.

The Question Analysis Module consists of two parallel sub-modules, as illustrated in Figure 2.3. The first module is for extracting the focus of the question, whereas the second module is for determining the most likely classification of the question (QClass) into the pre-defined set of question classes.

The focus indicates what exactly the given question is asking for, and what type of entity it is. In the example that is stated in Section 1.1, the focus is the collection

of these parts of the question: “ovasının adı” (name of a specific plain), because the question asks for a name. In particular, it asks the name of a plain. Note that, we refer to each word of the question as a “part”, as a part represents a single word in the question that has been annotated with extra information such as its morphological root, part-of-speech, dependency tag, and etc. Therefore, the phrase “ova adı” (name of a plain) can be constructed even syntactically from the phrase “ovasının adı” (name of a specific plain), since we already have the morphological roots attached to the question parts. Because “ova”(plain) is the root, and “sı” and “nın” are possessive adjuncts inducing together the meaning: “a name of a plain of ”. Moreover, the QClass for this question is ENTITY.PLAIN (see Table 4.2).

In the following example, the focus is the parts “denizci kimdir” (Who is the sailor), and the QClass is HUMAN.INDIVIDUAL. The rationale for the focus is the question asks for a person, and it is known that the person is a sailor. Observe that we omit the distinctive properties of the entity in question (e.g. the first sailor), because at this point, we are mostly interested in “is a” and “part of” relations that indicate a certain type of the entity. The remaining properties are used by the subsequent modules of HazırCevap to semantically prune both the relevant knowledge units and the candidate answers.

“Dünyayı dolaşan ilk denizci kimdir?”

“Who is the sailor that first circumnavigated the Earth?”

In order to extract the focus and QClass, HazırCevap uses separate rule based experts for each question type that are armed with rules over the dependency parse tree of the question, along with a novel Hidden Markov Model (HMM) based algorithm that uses the serialization of the same dependency tree. The final result is computed using both expert confidences that are learned from the training data set, and probabilities that HMM produces for each part of the question to be a part of the focus. In parallel, QClass is extracted by using a weighted rule-based classifier, which extracts the coarse

class by manually constructed phrase-based rules of Turkish questions.

2.3. Information Retrieval Module

As discussed in Section 2.1, there are two main goals of the IR module; to search for the documents and passages that are relevant to the concepts and entities mentioned in the given question, as well as to catch some immediate candidate answers from the previously known facts and relations.

After the question analysis, the generated query is passed to the IR module. IR module has two different sub-modules responsible for structured and unstructured searches.

IR module performs the structured search by looking for the previously known facts and relations such as the previously defined relations like is_a{İstanbul, CITY} and automatically extracted relations like effects{climate, vegetation}. Previously collected facts and relations make possible relating the facts and entities mentioned in the given question and thereby inferring some immediate candidate answers as explained in Chapter 5 in detail.

The unstructured search is performed by querying the unstructured document base by Apache Lucene and Indri search engines. The top related documents and passages are then selected to be passed to further modules to search for candidate answers. Section 5.1 explains in detail the formulation of a query and the querying procedure.

3. DATA PREPARATION AND RESOURCE SELECTION

This study is conducted by a group of researchers from both Computer Engineering and Educational Technology Departments. The design, implementation and testing of HazırCevap system is performed with the data collected and prepared by Educational Technology researchers.

3.1. Determining Prototype Domain

The system's initial design is constrained to a limited field of research, which made possible that the individual modules to be clear enough to be easily designed and tested separately. In this regard, one course from the high-school curriculum is selected to be the prototype domain for the system.

Three prominent features required in the prototype domain is that firstly it should be a discipline applied to both middle and high school students, secondly most of the information in course resources have textual representations and lastly the subject of the course should be fairly objective. The information in different resources should be approximately uniform. Considering all these requirements, the History course may be considered as an appropriate prototype domain. However, the subjective characteristics caused by the different interpretations of historical events in the resources render history a erroneous choice. Chemistry, Physics and Mathematics are ruled out for the intensity of symbolic representations of the information which their resources include. The system needs to be able to easily analyze and extract useful information from the resources.

After the initial considerations, Geography course is chosen to be the prototype domain of HazırCevap. The fundamental reasons for choosing Geography include that it is a part of both middle- and high-school curricula and almost all information regarding Geography are based on objective geographical facts.

3.2. Factoid Questions vs. Open-Ended Questions

Aside from the domain, QA systems have drastically different characteristics depending on whether the question is factoid or open-ended.

Factoid questions have a single, clear and objective answer that doesn't depend on the responder, such as the following question:

"Fransa'nın başkenti neresidir?"

"What is the capital of France?"

Whether it is computer that is responding or a human individual, the answer is always Paris.

On the other hand, open-ended questions aren't looking for a single answer, but an obscure information such as an insight, remark or an answer involving, however contracted, a bit of interpretation. For example, consider the following question:

"Güney Amerika'da iklimin bitki örtüsüne etkisi nelerdir?"

"What are the effects of the climate to flora in South America?"

To answer this question, first of all the system needs to have a sense of "effect". Even if we assume that the system has the necessary information regarding "climate" and "flora", it also needs to extract the complex relationships between them and some sub-concepts like "cold", "rain" and "tree", "flower" or "grass". In addition, the system needs to eliminate from all kinds of such effects the ones that can be found in South America. In order to perform such answering, the system needs to have a strong computational architecture to be of practical use, as well as a deep resource base. Because if the system is designed to answer open-ended questions, even if it is a

closed-domain system, it should have the hardware of an open-domain system, along with a strong competence on traversing through a map of concepts (e.g. ontology) to discover information between sub-concepts that would normally seem utterly unrelated to a factoid system. Most importantly, one of the most difficult problems of building such a system, is evaluation. Evaluating the end-to-end performance or the accuracy of a system with factoid questions is considerably easy, because there is a single right answer. However, open-ended questions have usually no answers with absolute truth or falsity, since they are open to interpretation. For example, for the open-ended question above, one may only list some of the effects of the climate to the flora in South America, while another write a whole composition about it, and they may both be correct. This fact renders the evaluation of such a system critically hard.

Contrary to the systems answering open-ended questions, the systems answering factoid questions are somewhat easier to design and implement in practice, since the information to be found as the answer is not open to any interpretation. A strong searching and information extraction capabilities, along with a considerably sized resource base are sufficient to find and extract a single answer with a controllable error threshold.

For numerous reasons such as controllable error threshold, easiness of evaluation and practical use, only factoid questions are considered for HazırCevap. The system is left to be improved further to open-ended questions in future studies.

3.3. Determining Resources

Firstly, the books provided by Turkish Ministry of Education in Educational Informatics Network[4] for 9, 10, 11 and 12 grades are added to the resources. These books are considered as primary resources for the system, because they cover all the materials designed and provided as the curriculum for these grades again by the Ministry of Education. Using these books with a broad spectrum of content, 1000 factoid questions from the Geography domain are collected and prepared for processing. The most

[4]http://www.eba.gov.tr

important aspects of these questions include that they cover a wide range of topics, they have different grammatical structure and they are strictly factoid, in other words there should be only one and consistent answer for each and every question.

A set of 200 questions is selected as a representative of the whole set. The selection is performed by considering the question class (see Table 4.2), topic and grammatical structure. These questions are then used to evaluate the individual web pages to be appended to the resource base of the system. Each question is passed to Google engine to produce the initial set of web pages that will be considered as addendum to the primary resource base. Only the web pages containing the right answers to the questions are considered. 18 web pages are determined this way and formed the initial set, which is never subjected to any analysis other than to contain the right answer. Table 3.1 shows the complete list of such web sites. These web sites are evaluated according to the confidence and coverage metrics developed by educational technology researchers. The top 4 web sites with highest scores are selected to be among the resources and included to the system.

3.3.1. Confidence Metric

The confidence metric that is used to evaluate the reliability of a web site is adapted from the online "Web Site Evaluation Checklist[5] " provided publicly available by the University of California, Berkeley. This metric is developed by the Berkeley university library with the purpose of determining reliable web resources. The metric is translated and transposed to Turkish according to the needs of HazırCevap system. After the adaptation, the metric includes four main criteria with ten sub-criteria. Besides the qualitative criteria, a scoring system is also developed. Each site is scored with 1 (full), 0.5 (half) and 0 (none) points for each sub-criteria. Full points indicate complete confidence, half points indicate confidence but there are some obvious deficiencies, and no points suggest that the site is unreliable at all.

"Accuracy and proficiency" criterion measures the general reliability of the web

[5]www.lib.berkeley.edu

Table 3.1. Initially Collected Web Sites.

1	tr.vikipedia.org
2	www.cografya.org
3	www.bilgiustam.com
4	www.cografya.gen.tr
5	www.cografyalar.com.tr
6	www.mgm.gov.tr
7	www.msxlabs.org
8	www.diyadinnet.com
9	www.bilgibirikimi.net
10	www.cografyamiz.blogcu.com
11	www.konuanlatimi.gen.tr
12	www.nedirnedemek.com
13	www.hakkinda-bilgi-nedir.com
14	www.forumdas.net
15	www.turkcebilgi.com
16	www.cografyaegitimi.biz
17	www.acikders.org.tr
18	www.bilgizenginleri.com

site. Firstly the domain name is considered. The domain names having "gov", "k12" or "edu" extensions are automatically considered as reliable, since they are prepared by the organizations which are in concordance with the government. The publisher denotes if the web site is a personal site and the existence of an "About" page indicates the professionality of the content. Personal web sites or blogs are considered to be harmful to the system, since individuals may alter the content potentially in any inconceivable way. Additionally, the existence of "About" pages is treated as a separate criterion, since the information about the people who have created the site is greatly helpful in considering the reliability of the web site.

"Purpose and scope" criterion is about the content and the organization of in-

Table 3.2. Web Sites Confidence Evaluation.

URL	Accuracy	Purpose	Design	True Answers	TOTAL
www.acikders.org.tr	4	3	1	1	9
www.mgm.org.tr	4	1.5	1	1	7.5
www.cografya.gen.tr	3.5	2	0.5	1	7
tr.vikipedi.org	2	3	1	1	7
www.diyadinnet.com	3	1.5	0.5	0.5	5.5
www.bilgiustam.com	1	2	1	1	5
www.nedirnedemek.com	2	1.5	0.5	1	5
www.turkcebilgi.com	1	1.5	1	1	4.5
www.cografyamiz.blogcu.com	0.5	3	0.5	0.5	4.5
www.konuanlatimi.net	1	1	0.5	1	3.5

formation within the web site. The existence of complementary and annotating links withing the content increases the reliability of the web site, and vice versa. For example, a link to a web site about dining within the Geographical textual content indicates a deficiency and decreases the reliability of the content. Furthermore, it is also important that all the links are functional, in other words the broken links also decrease the reliability score. Additionally, any reference to a scientific research indicates that the content is indeed credible.

"True answer" criterion is added to the metric, since it is of utter significance for a QA system to find the true answers within the resources. With this metric, it is measured that for a given question, a consistent and true answer must be found within the web site's content, explained in an objective manner. It is possible that the site contains only partial answers. Furthermore, it is also possible that even if the web site is not about geography per se, it answers some geography questions with single and clear answers. Therefore this metric is used to measure these discrepancies with the view of objectivity.

"Design" criterion tries to measure the reliability of a web page considering the overall design of the web site and the representation of information. At first glance, this criterion seems rather subjective and qualitative in measuring web site's reliability.

However, it has a strong indication on the professionality and objectivity of the authors as well as the content itself. Therefore, the professionality and meticulousness of the design is considered to be significant in reliability metric.

The 18 web sites shown in Table 3.1 are evaluated according to these criteria and ten sites with highest confidence scores are determined as illustrated in Table 3.2.

3.3.2. Coverage Metric

In addition to the confidence metric, a coverage metric that measures how well a website covers the topics of our prototype data set is developed. Using a supplementary search engine, every web site is tested against the sample data set containing 200 questions representing the whole 1000 question data set. In particular, the metric tests if a web site contains the true and objective answer of a given question. Therefore, the percentage of questions that are correctly answered by a web site determines the web site's coverage.

Table 3.3. Web Sites Coverage Evaluation.

Web Site	**Coverage Score (200)**
www.msxlabs.com	173
www.turkcebilgi.com	162
tr.vikipedi.org	161
www.cografya.gen.tr	147
www.diyadinnet.com	146
www.bilgiustam.com	136
www.forumdas.net	136
www.konuanlatimi.gen.tr	132
www.hakkinda-bilgi.nedir.com	130
www.bilgizenginleri.com	127

The top ten web sites having the highest coverage are determined by computing and ranking their individual coverage scores against the sample set of 200 questions.

These web sites are listed with their coverage scores in Table 3.3.

Finally, the top ten web sites having the highest confidence and highest coverage scores are matched such that the final mix of web sites are optimally reliable and answer most of the questions in the data set. For this purpose, four web sites which are listed in both top ten lists are selected to be a part of system resources, as listed in Table 3.4.

Table 3.4. Selected Web Sites to Extend System Resources.

Web Site
www.cografya.gen.tr
tr.vikipedi.org
www.diyadinnet.com
www.bilgiustam.com

3.4. Resource Compilation

The web sites and pdf documents are subjected to a set of pre-processing stages in order to be uniformly integrated to the system as a single primary resource base.

For the preparation of web pages, firstly the web sites are dumped with both their structure and content into the system server and scanned page-by-page in order to extract useful content from their source codes. For example, the title of the resource document that will be generated by this page is obtained from the first part of the web page, in particular from the <title> ... < /title> tags. The title of the document is updated when any other descriptor header such as <h1> ... < /h1> is encountered while processing the main content part, <body> ... < /body>. The main content is obtained by the textual chunks usually contained within <p> ... < /p> tags while processing the body. This way, every page in a selected web site is turned into a titled document to be indexed and searched by the search engines.

Similar strategy is employed for pdf documents as well. Firstly, textual content is extracted from the pdf documents using an external software, titled Fine Reader.

Analyzing this textual content, the big document is divided into smaller documents each having its own title based on its contents. For example, if the pdf document is a book, each section becomes a document having a title compiled from the header of the section and the containing chapter. This way, specialized small documents are obtained from the big pdf document.

Regardless of whether the document is originated from a pdf document or a web page, the documents with the same titles are appended together to avoid any duplicity. In the end, system's primary resources are compiled into a single big set of titled documents.

3.5. Question Data

One of the major contributions of this study is to provide a gold standard, diverse set of Turkish questions from the prototype domain of Geography, manually annotated by human experts. The data set contains 100 instances in the following format:

{Question Text | Focus Parts Text | Coarse Class | Fine Class | Answer }

Approximately 30 per cent of the data set consisted of actual questions posed by teachers, collected from textbooks related to Geography and online materials. The rest were generated by three of the researchers, who are educational technologists, based on actual Geography texts used in grades 9 to 12 in high schools in Turkey.

Table 3.5. Inter-annotator Agreement Scores for Focus and QClass.

Focus Agreement	% 82
QClass Agreement	% 92

We made use of two strategies in data annotation: focus annotation and QClass annotation. Three researchers (two of whom are educational technologists) manually identified the focus in each question, while two researchers (one educational technologist) annotated the questions for QClass. The evaluations were later compared to the developer's judgment. The inter-annotator agreement scores are reported in Table 3.5.

4. QUESTION ANALYSIS

As explained in Section 2.2, question analysis tries to extract preliminary information from the given question useful to the subsequent modules in the system. In particular, question analysis extracts the focus of the question, along with the classification of the question (QClass) into pre-defined set of question classes. This chapter discusses the methodology used by two sub-modules of the question analysis module responsible for focus and QClass extraction.

In addition to the related studies discussed in Section 1.2, one study draws attention regarding the focus extraction. This study, perhaps the study that is most relevant for our own question analysis methodology is conducted by [26], where rule-based and statistical methods are utilized together to extract the question focus in an open-domain QA system. In this study, a binary classification using Support Vector Machines (SVM) is performed on words in the English questions that are parsed using a constituency parser. Further, experts with manually tailored rules are used to identify the different features which are then deployed in the SVM. In contrast, our analysis separately uses both rule-based and statistical models to extract the focus. It also performs question classification for Turkish questions that are parsed using a dependency parser. Additionally, a sequence classification is performed using a Hidden Markov Model (HMM) based algorithm, whose results are then combined with the results of the rule-based experts to produce the final focus. Unfortunately, our study is incompatible for comparison with this study. Firstly, because the definition of the focus in [26] depends on a constituency parser and a co-reference resolver, which currently do not exist for Turkish. Therefore, it is neither possible to define equivalent rules for the English data set, nor to apply the techniques proposed in [26] to the Turkish data set.

4.1. Focus Identification

For *focus* extraction, we have a fastidious rule based focus extractor, , the *Distiller*, with specifically tailored rules over the dependency trees for almost all types of factual questions in the Geography domain, and an HMM based classifier, , *HMM-Glasses*, which uses a variation of the Viterbi algorithm [27] that essentially renders it somewhat more liberal than the Distiller to a certain extent. Other than one common trait, that is operating on the dependency relations among the words in the question, their approaches to the main problem (i.e. to extract the focus) are based on completely different principles in different levels of resolution. This distinction is critical to our methodology, since it provides the necessary insight for the model to efficiently handle languages with rich derivational structure, such as Turkish. At this point, a delicate balance is required for the combination of these models. For this purpose, we take into account the individual confidences of both the Distiller and HMM-Glasses, rendered through their individual performances over the training data set. Additionally, for the classification of the question into predetermined classes from a certain domain (Geography in our case), we have a rule-based classifier, which extracts the coarse class by manually constructed phrase-based rules.

4.1.1. Distiller

We observed that in our selected domain of Geography, there are certain patterns of question statements (based on the predicate), common to the majority of the questions. We identified each such pattern (*question type*) and defined manually sets of rules (*experts*) for the extraction of the focus from the dependency parse tree of each question.

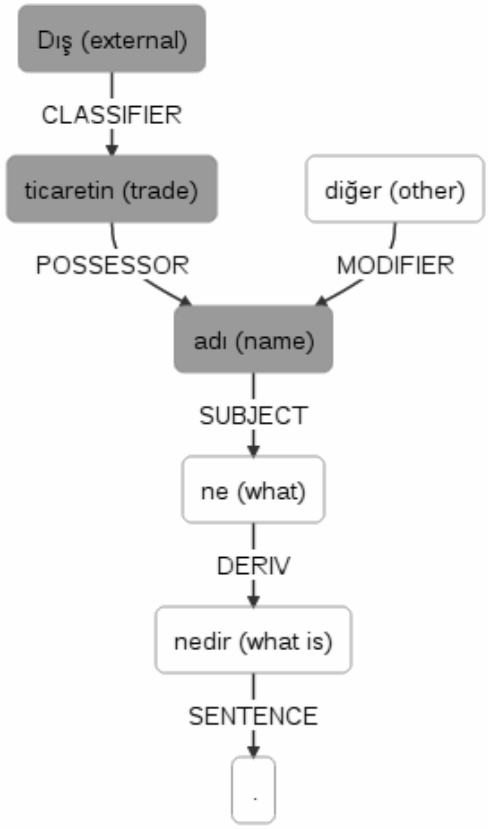
Currently we have seven rule-based experts, along with a generic expert that handles less frequent cases by using a single generic rule. The primary reason of the inclusion of a generic expert is data scarcity. However, we prefer to make it optional, because having a specific general expert along with a finite number of experts may result in a penalized precision as opposed to more or less increased recall, depending

on the data set size, which may not always be a desirable option in practice. All experts and their question frequencies in the data set are given in Table 4.1.

The rules contain instructions to navigate through the dependency tree of a given question. For example, the rule for the "nedir" *(what is . . .)* expert, and the rule for the "verilir" *(. . . is given . . .)* expert, as well as the generic rule are as follows (examples provided in Figure 4.1).

nedir: *(what is . . .)*

- Grab the *SENTENCE* in the question
- Grab and traceback from the *SUBJECT*, and collect only *POSSESSOR* and *CLASSIFIER*



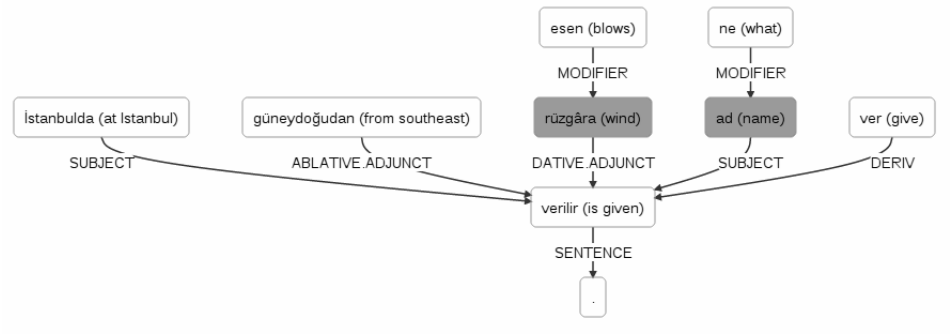
Dış ticaretin diğer adı nedir?

What is the other name of external trade?

Figure 4.1. *nedir* expert reports that the focus is a "name of external trade".

verilir: *(. . . is given . . .)*

- Grab the *SUBJECT* of the *SENTENCE* in the question
- Grab and traceback from the first degree *DATIVE.ADJUNCT* of the *SENTENCE*, and collect only the first degree *MODIFIER*



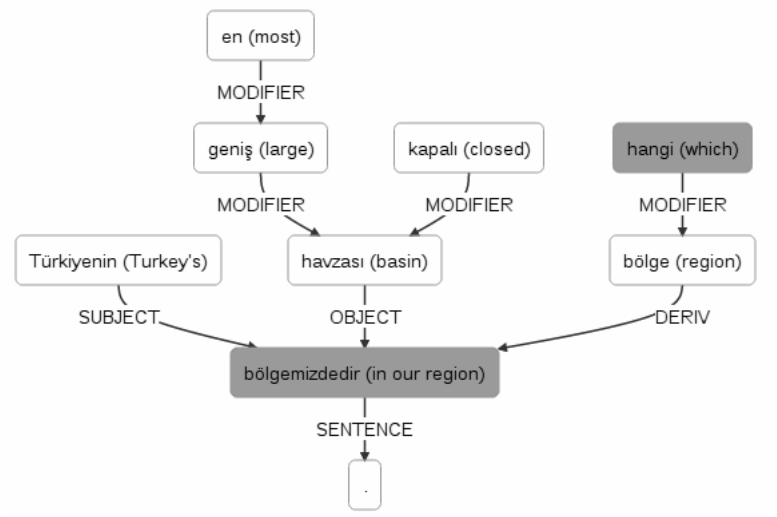
İstanbulda güneydoğudan esen rüzgara ne ad verilir?

What is name of the wind that blows from Southeast in Istanbul?

Figure 4.2. *verilir* expert reports that the focus is a "name of a wind".

hangi*(. . . which . . .)*

- Grab the *MODIFIER* "hangi" in the question
- Trace forward to *SENTENCE* and grab all parts within (without *DERIV* parts)



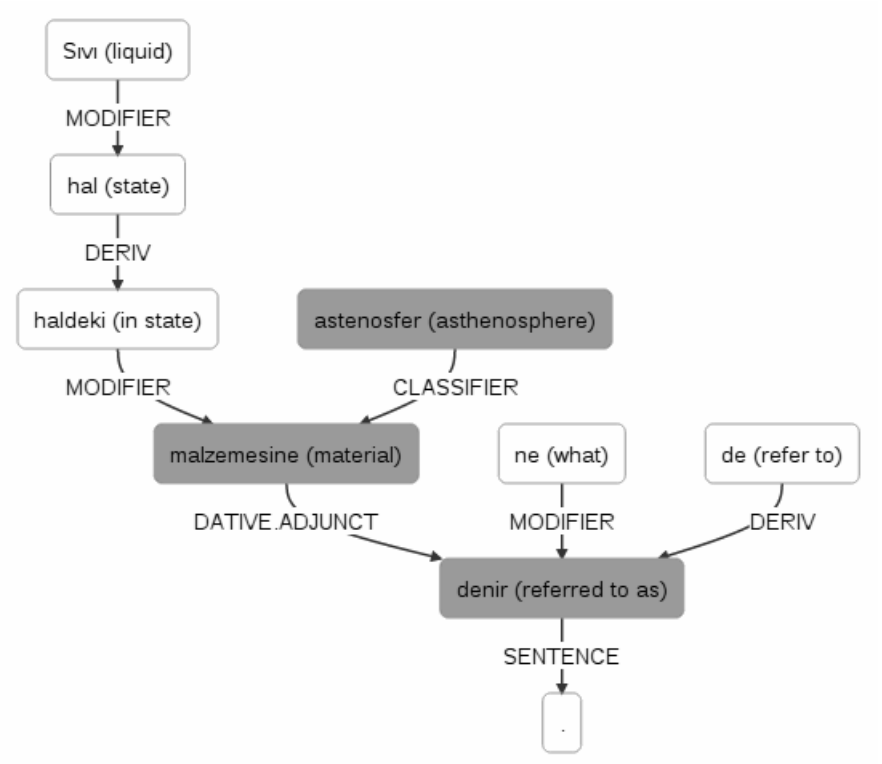
Türkiye'nin en geniş kapalı havzası hangi bölgemizdedir?

On which of our regions located the largest basin of Turkey?

Figure 4.3. *hangi* expert reports that the focus is a "region".

denir *(. . . referred to as . . .)*

- Grab the *SENTENCE*, along with the *DATIVE.ADJUNCT* of the *SENTENCE*
- Traceback from *DATIVE.ADJUNCT* and collect all (if any) *CLASSIFIERS*



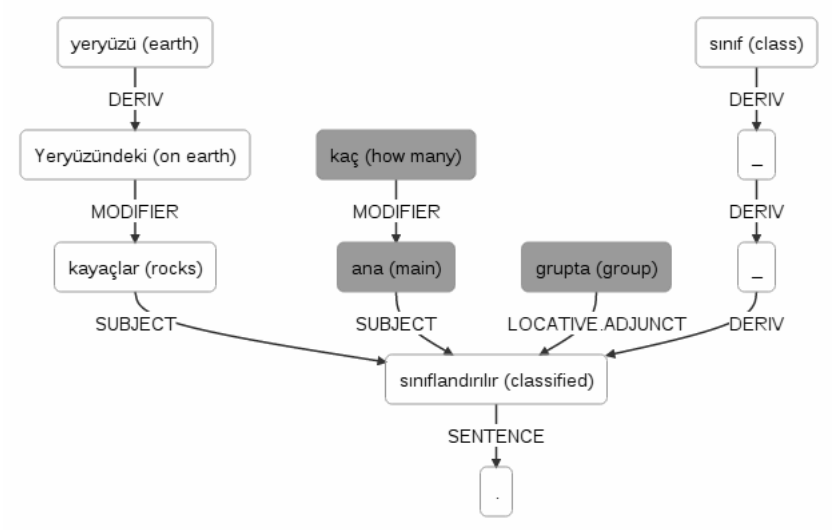
Sıvı haldeki astenosfer malzemesine ne denir?

What is referred to the asthenospheric material in liquid form?

Figure 4.4. *denir* expert reports that the focus is a "referral to an asthenospheric material".

kaç *(how many . . .)*

- Look at the parent of the *MODIFIER* "kaç"
- If the parent is *SENTENCE* (or a *DERIV* of *SENTENCE*), grab the *SENTENCE* along with "kaç"
- If the parent is *SUBJECT*, grab "kaç" with *SUBJECT* along with the *LOCATIVE.ADJUNCT* of the *SENTENCE*
- Else, look for a *SUBJECT* on the branch of "kaç" *MODIFIER*, grab the whole branch from *SUBJECT* to *SENTENCE* (including "kaç").



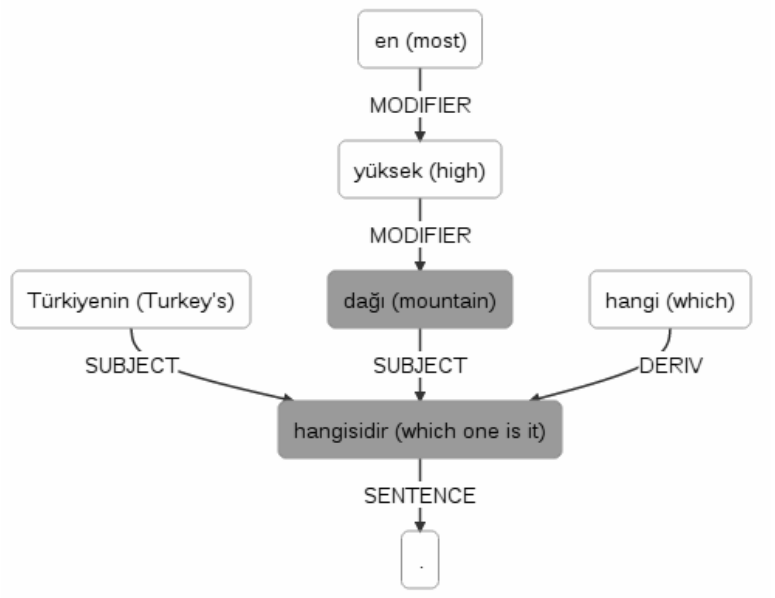
Yeryüzündeki kayaçlar kaç ana grupta sınıflandırılır?

How many main groups are the rocks on Earth classified into?

Figure 4.5. *kaç* expert reports that the focus is a "number of main groups".

hangisidir *(which one is it . . .)*

- Grab the *SUBJECT* of the *SENTENCE*
- Traceback from *SUBJECT* and collect all *CLASSIFIER* and *POSSESSOR* parts.



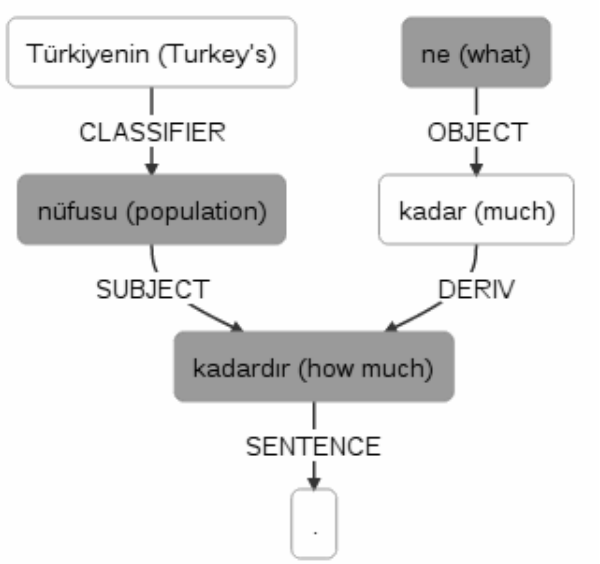
Türkiye'nin en yüksek dağı hangisidir?

Which one is the highest mountain of Turkey?

Figure 4.6. *hangisidir* expert reports that the focus is a "mountain".

kadardır *(how much ...)*

- Look for a *OBJECT* and a *SUBJECT* part of the *SENTENCE*
- Traceback from found parts and collect all *CLASSIFIER* parts.



Türkiye'nin nüfusu ne kadardır?

How much is the population of Turkey?

Figure 4.7. *kadardir* expert reports that the focus is a "population value".

generic:

- Grab the *SUBJECT* of the *SENTENCE* in the question
- Traceback from the *SUBJECT*, and collect the first degree *POSSESSOR* and/or *CLASSIFIER*, along only with their *POSSESSOR* and/or *CLASSIFIER*

Every rule-based expert has a confidence score based on its performance for extracting the correct focus parts from the questions belonging to its expertise. This score is used to indicate the reliability of the expert's judgment later when combining its result with the *HMM-Glasses*. The confidence scores, along with the focus parts of a question Q are reported by both the *Distiller* and the *HMM-Glasses* in the format of triplets:

$$\langle fpt, fpd, fpc \rangle_n$$

Table 4.1. Experts and their question frequencies in the training data.

Expert Type	***Frequency (%)***
generic	25.6
hangi *(... which ...)*	19.5
nedir (what is ...)	15.0
denir *(... referred to as ...)*	9.6
kaç *(how many ...)*	9.6
verilir *(... is given ...)*	7.2
hangisidir *(which one is it ...)*	7.2
kadardır *(how much ...)*	6.3

where $n \in |Q|$ [6] , fpt stands for *focus part text*, fpd is *focus part dependency tag* and fpc denotes *focus part confidence* score. Both models produce such triplets for each focus part that they extracted. However, there is a significant distinction in the way that the confidences are reported for each part of the extracted focus between the rule-based and the statistical models. As explained in detail in Section 4.1.2, *HMM-Glasses* work on individual parts of the question, while the *Distiller* extracts sub-trees from the dependency tree of the question. Therefore, the *Distiller*'s resolution is not big enough to consider the individual probabilities for each part to be in the focus. Thus the *Distiller* produces a collection of parts as the *focus*, along with a single confidence score (*total confidence score*) reported by the expert in charge, this is mapped to fpc scores of all parts, rendering all parts in the focus equal from the *Distiller*'s perspective.

4.1.2. HMM-Glasses

HMM-Glasses models the focus extraction as a HMM and performs a sequential classification on the words in the question using the Viterbi algorithm. Having only two hidden states, namely *FOC* (i.e. the observed part is a focus part) and *NON* (i.e. the observed part is not a focus part), it treats each question part as an observation, and decides whether the observed part is a part of the focus of the question.

[6] $|Q|$ denotes the number of words in the question Q

We first serialize the dependency tree of the question and feed the algorithm the serialized tree. Serialization (or encoding) of a tree is to systematically produce a sequential representation of it, which is mostly employed in the fields of applied mathematics, databases and networks [28, 29]. Evidently the method with which the tree is serialized has an observable influence on the characteristics of the algorithm's results. We investigated this effect with two general serialization approaches, empirical tests and results are reported in Chapter 6. Common approaches in tree serialization try to efficiently serialize the tree within the information theoretical resource bounds (in terms of time and space), while taking into account also the deserialization process [30]. On the other hand, we are only concerned with the coherency of the tree structure. In other words, the dependency relations should be consistent among all the serialization methods. Therefore, we considered the simplest possible methods, *forward mode* and *backward mode*.

While constructing the sequence from the dependency tree in forward mode, left children (according to the reverse visualization of the dependency tree) take precedence over the right children to be taken into the sequence. Therefore, the left-most branch is taken first, then the branch on its immediate right is taken, and so on. Finally the parent is added. Backward mode is simply the other way around, where the right children take precedence over the left children. Any difference in serialization changes the whole learning process, thereby renders the learned features unique to a particular serialization. This therefore provides a noticeable diversity in the characteristics of the learning, depending on the serialization method. Below are the serializations of the question in Figure 4.1. Recall that we only consider the morphemes of the words (i.e. stripped from all the adjuncts).

forward serialization (->)

Dış	ticaret	diğer	ad	ne
(external)	*(trade)*	*(other)*	*(name)*	*(what)*
FOC	FOC	NON	FOC	NON

Essentially, forward mode serialization corresponds to reading the question from left to right (or start to end), while backward mode corresponds to reading it from

backward serialization (<-)

ne	ad	diğer	ticaret	Dış
(what)	*(name)*	*(other)*	*(trade)*	*(external)*
NON	FOC	NON	FOC	FOC

end to start. Different serialization approaches potentially allow ensembles of various kinds of models, handling different parts of the question as they have learned different features of the data while training. Therefore, a more complex model can be obtained by combining multiple *HMM-Glasses* having different serialization approaches.

We model the focus extraction problem as a HMM by firstly computing the prior probabilities of our hidden states (i.e. *FOC* and *NON*), and secondly learning the probabilities from the given set of serialized questions as follows:

$$a_{jk} = P(t^j|t^k) \qquad b_{ij} = P(w_i|t^j) \tag{4.1}$$

where a_{jk} represents the probability of being in state t^j given the previous state is t^k, and b_{ij} indicates the probability that the current observation is the word w_i given that the current state is t^j. Decoding is performed using the Viterbi algorithm, where the states correspond to the nodes in the produced Viterbi path indicating the most likely judgments for each part to be a focus part of the question. Further, the observation probabilities b_{ij} are used as confidence scores (i.e. fpc) in the triplets. Recall that all results are reported as triplets (see Section 4.1.1).

In all parts of the question analysis, taking advantage of the dependency relations among the words in the question whenever possible has prominent benefits, compared to mere syntactic approaches for languages with a rich derivational structure, where for instance possible long distance relationships in the question statement can easily be determined. Therefore, the very first design of the *HMM-Glasses* was planned

to learn and evaluate the dependency tag sequence of a question, which essentially corresponds to learning the tree shape, rather than the sequence of words. However, this approach mislead the model, as there are some tags that occur more frequently in questions than others, as for example a question often has only one *SENTENCE* tag, while it has lots of *MODIFIER* tags. More importantly, the focus is often a small part of the question. Thus, for example, the judgment of whether a *MODIFIER* part is a focus part is strongly biased by the fact that the number of cases a *MODIFIER* is a NON will be orders of magnitude higher than otherwise. Furthermore, working with the normalized frequencies requires a lot of training data for the model to have a statistically significant learning experience. Therefore, *HMM-Glasses* currently learns the probabilities of the part texts (i.e. words) in the question. This leaves the model with no dependency relation information at hand. However, it is compensated by the *Distiller* as the experts use by definition only the dependency rules for extraction.

4.1.3. Combination of Distiller and HMM-Glasses

Recall that the *Distiller* outputs the focus parts with a single total confidence score of the expert that produced the results. In addition with the part-wise confidences that *HMM-Glasses* produces, we have:

$$\left\{\begin{array}{cc} \text{HMM} & \text{Distiller} \\ \langle \text{fpn}_1,\, fpt_1,\, fpc_1 \,\rangle & \langle \text{fpn}_1,\, fpt_1,\, fpc \,\rangle \\ \langle \text{fpn}_2,\, fpt_2,\, fpc_2 \,\rangle & \langle \text{fpn}_2,\, fpt_2,\, fpc \,\rangle \\ \vdots & \vdots \\ \langle \text{fpn}_p,\, fpt_p,\, fpc_p \,\rangle & \langle \text{fpn}_q,\, fpt_q,\, fpc \,\rangle \end{array}\right\}$$

Combination of the candidate focus parts produced by different models is performed in a part-wise manner. In other words, models try to convince each other about each part being among the final focus parts. To do this, we make use of the fpc scores, weight them with the models' individual f-scores over the training data and grab the maximum. Note that, if a part is determined as a candidate focus part by only one of the models M_1 (i.e. the other model M_2 predicts that this part is not a focus part), then we compute the confidence score of M_1 as described above and compare it with

the f-score of M_2. If the confidence score of M_1 is greater than that of M_2, the word is classified as a focus part, otherwise it is excluded from the focus.

The empirical results regarding both the individual and in combination of the developed models for focus extraction are reported and discussed in Section 6.1.

4.2. Question Classification

For question classification, we manually pre-determined two types of classes, namely coarse and fine classes, adapted from [9, 10], with different semantic resolutions. A question's fine class establishes a strong link to the specific domain at hand, while its coarse class essentially introduces a generality into the model that would render the classification applicable in domains other than Geography.

Table 4.2. Coarse Classes for the Geography Domain.

Question Class	***Frequency (%)***
DESCRIPTION	25.2
NUMERIC	24.2
ENTITY	19.6
TEMPORAL	12.4
LOCATION	11.9
ABBREVIATION	3.8
HUMAN	2.4

Currently we have seven coarse classes, along with a total of 57 fine classes. In this study, we only concentrated on coarse classes. We plan to perform classification of fine classes using statistical approaches, which requires comprehensive number of questions in each fine class. All coarse classes and their question frequencies are listed in Table 4.2.

In order to classify a given question into one of the coarse classes, we devised a set of common phrases for each class unique to that class. For example, for the class

NUMERIC, we have two phrases: "kaç" *(how many/how much)* and "kadardır" *(this much/that many)*. The classifier searches for these patterns in a given question and classifies accordingly. Sample phrases for each class is listed in Table 4.3.

Table 4.3. Sample Phrases for Coarse Classes.

Coarse Class	**Sample Phrases**
DESCRIPTION	"ne isim verilir", "temel sebebi nedir"
NUMERIC	"kaç", "ne kadar"
ENTITY	"rüzgar tipi", "dağı hangisidir", " hangi ova"
TEMPORAL	"hangi tarihte", "kaçıncı yüzyıl"
LOCATION	"hangi bölge", "nerede", "nereye"
ABBREVIATION	"açılım", "kısa yazılışı"
HUMAN	"kimdir", "kimin", "hangi kültür"

We additionally implement a statistical classifier that employs a tf-idf based weighted bag-of-words strategy, as a baseline model to compare with the rule-based approach. In baseline model the weight of a word w for a class c is computed as follows.

$$\text{tf-idf}_{w,c} = tf_{w,c} \times idf_w \tag{4.2}$$

where $\text{tf}_{w,c}$ indicates the number of times word w occurs in class c, and idf_w is computed as shown below.

$$idf_w = \log \frac{\#\text{ of classes}}{\#\text{ of classes containing w}} \tag{4.3}$$

Then, for a given question Q, we assign it to the class that maximizes the sum of the tf-idf scores of the words in the question:

$$\arg\max_c \sum_{w \in Q} \text{tf-idf}_{w,c} \tag{4.4}$$

The empirical test results of the developed techniques for question classification are reported and discussed in Section 6.2.

5. INFORMATION SEARCH

For every question answering system, some type of information retrieval (IR) needs to be performed to search for an answer or information that will lead to an answer. Therefore, having a powerful IR module is critical for any question answering system. The relationship between an information retrieval system and a factoid question answering system is empirically shown in [31]. The design and implementation of an IR module of an answering system should be in concordance with the purpose, domain, along with its usage and even the profile of the users [32].

Previous studies shows that one of the most important design decisions of systems that have its own resources and perform the querying with its own customized queries and specialized methods, is whether to keep the resources in structured or unstructured manner. Therefore, the system resources are divided into two distinct groups based on their structures and querying methods, namely structured and unstructured [2].

Structured resources are compiled by extracting certain kinds of information from the previously designated textual resources. This extraction can either performed automatically or manually (e.g. by crowd-sourcing). The resources of systems that based on an ontology for example, are single or several big ontologies, often constructed by crowd-sourcing. These ontologies may be specifically constructed for a particular system. Yet some systems use external general purpose ontologies to employ general semantic information in answering methodologies [33]. Most common such ontologies are YAGO (Yet Another Great Ontology) [34] and WordNet [35]. With the help of these ontologies, a question answering system specialized in finding relations between words and concepts can find answers with great success. Structured resources that are not based on an ontology are built by a special configurations using a pre-defined formats. The most common formats in this regard are <Subject,Verb> or <Verb, Object>. Because of this formatting for example the relation extraction component mentioned in Section 2.1 is very important to match these structured relation information with the ones in the given question.

Systems that use unstructured resources on the other hand, try to find relevant documents and passages that are likely to contain the answer, using textual pattern recognition by rule-based and statistical passage retrieval algorithms such as *bm25*, *MultiText* or *SiteQ* [36]. Further, some systems even search for exact phrases using named entities and different types of phrases such as noun, verb or prepositional phrases [37]. However, systems having a broader domain and answering to lots of question types often employ a general purpose search engines with automatically generated customized queries from the given question. The most common search engines in this regard are open source engines *Indri* [38], and *Apache Lucene* [39]. IBM's Watson for example, employs the combination of Apache Lucene and Indri with highly specialized query building system [40], along with a large-scale lexicalized relation extractor to process massive amounts of text and produce aggregate statistics to help with candidate answer generation and type coercion on the generated candidates [19].

In HazırCevap, we employ a similar strategy with Watson. We have Apache Lucene and Indri search engines working in parallel to retrieve relevant documents and passages from unstructured textual resource base. At its current stage, however HazırCevap doesn't have a structured resource search, it does have over two million Turkish sentences dependency parsed and ready to be processed, as well as the processor programs that can read and decompose such parsed sentences. Therefore, we safely left the plan to expand the capabilities of the IR module of HazırCevap in the future.

Unstructured resources contain titled documents of textual contents that are not annotated, parsed or altered in any other way. Determining the related documents depend on the complex expressions of the relations between the given question and the resource documents in the retrieval model. Therefore, customizable search engines (through their query language) can be made to perform the search in a very different perspective with different configurations. Furthermore, the difference in the core retrieval models of different customizable search engines may provide even wider perspective in search when they are used in a combination in the system. For this reason, the unstructured search in this study is performed using Apache Lucene and Indri search engines together with the same query formulation and scoring methods,

in order to increase information retrieval recall for the overall system.

The retrieval model of Indri search engine includes combinations of language models and inference networks. Having its own query language, Indri can index widely used file formats such as pdf, html and trec, and searches directly on them. It can work with terabytes of data, with utf-8 support and a considerably powerful API for lots of programming languages. Therefore it has being used in the last decade by lots of researchers around the globe for various fields. Additionally, it is also considered as a fine tool for languages that are harder to process than English, such as Turkish [38].

Apache Lucene (Solr) on the other hand, unlike Indri, uses tf-idf measure to resolve the relatedness in its core retrieval model. Although it doesn't have a query language per se, Apache Lucene has lots of different combinable query types, such as phrase query and proximity query, that make possible complex logical combinations to be constructed. Some of the most prominent features of Apache Lucene are low memory consumption, field specific search (title, link, etc.) and querying several indices simultaneously. Hence it has become very popular in the last decade in research as well as in the industry [39].

As illustrated in Figure 5.1, after formulating and scoring a query, HazırCevap feeds this query to Indri and Apache Lucene. The top ten documents and passages from both Indri and Lucene are then considered by the system as relevant documents and passages that most likely contain the answer of the given question.

5.1. Formulating A Query

One of the most important aspects of using search engines to retrieve relevant documents is the query formulation. The formulation of a query for search engines involves selecting and expanding the terms that are included in the query.

Term selection is highly critical in formulating a query, since further analyses such as expansion are based on the selected terms. Therefore, the terms that are chosen to

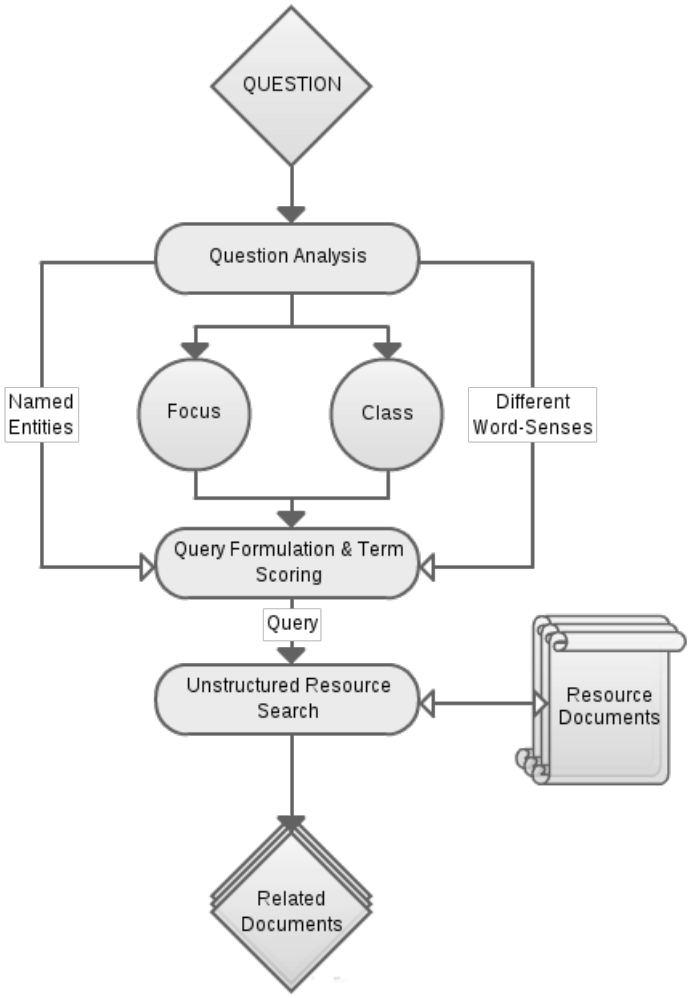


Figure 5.1. Information Retrieval Module.

be added to the query should be informative, as well as the number of terms is required to be as minimal as possible. The empirical effect of the quality of the term selection to the retrieval is reported in Section 6.3.

HazırCevap firstly selects all the terms in the question to form a query, and removes the stopwords and question words (e.g. "kim" *"who"*) as well as the punctuation marks. For example:

"Tsunami denen dev dalgalara ne sebep olur?"

"What causes the huge waves called Tsunami?"

The terms in this question is fairly clean, in terms of punctuations or word derivations. Firstly, all the terms are selected (without punctuations):

Tsunami - denen - dev - dalgalara - ne - sebep - olur

Then the question words are removed ("ne"(*"what"*)):

Tsunami - denen - dev - dalgalara - sebep - olur

Stop words are also removed ("olur"(*"do/does"*)):

Tsunami - denen - dev - dalgalara - sebep

After this removal, the morphemes of the words (extracted by the parser when parsing) are selected:

Tsunami - de - dev - dalga - sebep

Finally, the phrases (word groupings) are identified by a bigram language model, trained on the textual resource base.

Tsunami - de - *phrase*(dev dalga) - sebep

At this point, we have terms used in the given question that are related to what the question is asking. One more information is used before the query term scoring process, which the *focus* of the question that is determined by the question analysis component of the system. For the example above, the focus is the word "sebep"(*"reason"*). We also mark that word as the focus.

Tsunami - de - *phrase*(dev dalga) - *focus*(sebep)

The actual is generated with this information as follows. All the terms are combined with combiner operator of the particular search engine. Phrases are grouped in the query such that they should appear in the resource document as they appear in question. Additionally, the focus terms are included as "MUST BE" terms in the query, since *focus* indicates what kind of information that the question is really asking for. Figure 5.2 shows an actual non-weighted query generated for the question above to be used by the Indri search engine.

```
<parameters>
<index>/home/caner/clean/indri-5.0/indexFile/</index>

<query>
<number>0</number>
<text>#combine(Tsunami de #1(dev dalga) #band(sebep))</text>
</query>
</parameters>
```

Figure 5.2. Sample Non-weighted Indri Query.

6. EXPERIMENTS AND RESULTS

One of the major challenges we face was not having a suitable baseline (from previous studies etc.) to indicate the actual hardness of the problem and the actual efficiency of our solutions. Therefore, we implemented a baseline model for focus extraction that identifies the words adjacent to a question keyword for certain proximity as focus parts. Moreover, a tf-idf based statistical baseline model that employs a bag-of-words strategy is implemented for question classification as well. All the results are reported as comparisons to these baseline models in Section 6.1 and Section 6.2.

Since the data on which our models are evaluated have been prepared in this course of study, we build our strategy of evaluation around the concept of hygiene, where we ensure two fundamental principles. Firstly, at any point and for each model, scores are obtained from the results produced for questions with which the model *never* crossed before. Secondly, for a reasonable comparison between the models, same scores are computed for different models with different settings using the *same* questions at each iteration of the evaluation.

To evaluate the *Distiller*, the rule-based experts are developed by using only the first 107 questions, that we had at the beginning. Therefore, the remaining questions are safely treated as test data, as there were no modifications done after having a larger number of questions.

Evaluations for all the models are performed using stratified 10-fold cross-validation over all the questions. The final results (i.e. precision, recall and f-score) for focus extraction are obtained by macro-averaging the individual results.

Recall that the *Distiller* has the option to enable and disable the generic expert, while the *HMM-Glasses* has *forward*, *backward* and *forward & backward* modes that calibrate the serialization of the dependency tree. All the different combinations of these settings for each model are separately evaluated both individually and in com-

bination, in each iteration of the folding process. The results for focus extraction and question classification are shown in Tables 6.1 and 6.2, respectively.

6.1. Focus Identification Results

Individual evaluation of the *Distiller* resulted in comparable precision scores along with lower recall scores (compared to the combined models). A notable outcome of the *Distiller* evaluations is the behavior of the generic expert. Results indicate that generic expert lowers the accuracy of the retrieved results (i.e. precision), while increasing the coverage (i.e. recall) of the model. However, the two effects do not compensate, as the results show that f-score of the *Distiller* with the generic expert enabled is higher than the one with the generic expert disabled.

Distinct evaluation of the effect of the serialization methods indicates that for forward and backward modes, the forward mode is slightly better than the backward mode considering the f-scores. Backward mode seems to increase the recall of any model to which it is included, however, f-scores indicate that this increase in recall is not useful, because it in fact lowers the performance of the combined models whenever it is included.

Table 6.1. Evaluation Results of All Models for Focus Extraction.

Model	***Precision***	***Recall***	***F-Score***
Baseline	**0.769**	**0.197**	**0.290**
Distiller (Generic Enabled)	0.714	0.751	0.732
Distiller (Generic Disabled)	0.816	0.623	0.706
HMM-Glasses (Backward Mode)	0.839	0.443	0.580
HMM-Glasses (Forward Mode)	0.847	0.495	0.625
HMM-Glasses (Forward and Backward Mode)	0.821	0.515	0.633
Combined (Generic Enabled, Backward)	0.734	0.841	0.784
Combined (Generic Enabled, Forward)	0.732	0.846	0.785
Combined (Generic Enabled, Forward & Backward)	**0.721**	**0.851**	**0.781**
Combined (Generic Disabled, Backward)	0.821	0.759	0.789
Combined (Generic Disabled, Forward)	0.818	0.765	0.791
Combined (Generic Disabled, Forward & Backward)	**0.802**	**0.776**	**0.788**

In general, although the individual accuracy of the models are reasonable enough, the increase in the coverage (recall) for all combined models, having both the *Distiller* and *HMM-Glasses*, compared to the individual recall scores indicate that the combination is useful, as it does not sacrifice the precision scores that we observe in individual evaluations, thereby increasing also the f-scores. Therefore, we can conclude that the models complement each other nicely.

6.2. Question Classification Results

Results show that exploiting the domain knowledge resulted in a significant success that a statistical baseline model could not get near. However, manually crafted set of rules are a big problem when changing the domain. Therefore, a statistical learner that will automatically learn these domain specific phrases is planned for further development, since it requires significant amount of instances for each class. This scarcity is also the reason we leave the identification of fine classes for a future study. Table 6.2 shows the macro-averaged precision, recall and f-score of coarse class identification of the rule-based classifier, along with the results of the tf-idf based baseline classification.

Upper section is baseline tf-idf based model, and lower section is rule-based model.

6.3. Experiments on Information Search

Experiments on information retrieval module are conducted by firstly compiling queries in different settings and configurations. Secondly using these queries, Apache Lucene and Indri search engines are ran on the whole document resource base. Thirdly, the resulting documents from running both engines are assembled together to form a single set of documents that are hypothesized to be relevant to the originally given question. Finally, the tests that measure the relatedness of this final set of documents to the given question indicate the success rates of the initially generated queries.

For every technique that is employed at each step of the overall query generation sequence, explained in Section 5.1, we produce a single query in which that particular

Table 6.2. QClass Identification Results.

Classses	***Precision***	***Recall***	***F-Score***
Description	0.662	0.908	0.764
Temporal	0.767	0.618	0.670
Numeric	0.801	0.758	0.776
Entity	0.100	0.025	0.040
Abbreviation	0.933	0.766	0.823
Location	0.759	0.212	0.312
Human	0.600	0.600	0.600
Overall	**0.660**	**0.555**	**0.569**
Description	0.874	0.732	0.797
Temporal	1.000	1.000	1.000
Numeric	0.995	0.911	0.951
Entity	0.603	0.817	0.694
Abbreviation	0.871	0.894	0.883
Location	0.944	0.880	0.911
Human	0.869	0.833	0.851
Overall	**0.879**	**0.867**	**0.869**

technique is isolated. In other words, for a single question, we generate multiple queries each of which having one technique removed. This way, we can explicitly observe the effect of a single technique to the overall query generation performance.

Our document base currently contains 225089 titled documents that are indexed by both the content and the title, thereby enabling the search to be performed on both the content and the title of the documents. Both search engines work on same kind of index files that are prepared with the default settings and configurations of their indexers. We take 5, 10 and 20 top results of the engines and combine them to produce the final set of 10, 20 and 40 documents.

The relatedness of the final set of documents to the given question is measured by

searching the pre-annotated answer in the documents. If one of the documents contains the answer, then the document set that is extracted by the search engines considered to be related to the question. The answer is annotated as a single word (or a single number representing year, height, etc.). Therefore the search for the answer in the documents is performed with multiple representations of the answer. For example, for the hypothetical answer "Ağrı'daki dağ", the relatedness is tested with the variations some of which are listed below:

- Ağrı Dağı
- Ağrıdaki dağ
- ağrıdaki dağ
- Ağrı'da olan dağ
- Ağrı'nın dağı

Table 6.3 shows the effects of each query generation technique to the overall information retrieval performance.

Table 6.3. Information Retrieval Experiment Results.

Configuration	**5**	**10**	**20**
Final IR performance	47.6	51.4	58.8
include question words	44.3	49.7	55.6
include stop words	45.2	49.5	56.2
use full words	46.4	50.2	56.7
disable phrase detection	43.3	45.7	53.8
optional focus	48.5	53.2	60.1

As explained before, each configuration in the experiment rules out only one of the techniques to see its effect in isolation. For example, in "include question words" configuration, all the techniques in query formulation are used except the question word removal.

We observe a certain performance loss when excluding individual techniques,

except for the focus. As an interesting result, we observe that the "optional focus" configuration increases the overall system performance. Recall that the focus parts of the question that the question analysis module is extracted are annotated as "MUST BE" in the formulated queries. In other words, the retrieved documents must contain the focus parts exactly as they are in the original question. Being optional, the system performance seems to be increasing. This result may mislead us to believe that the focus is not so useful in the end. However in reality, this result validates our original understanding of the focus. As stated in Section 1.1, we defined the focus of a question as the information that indicates a certain type or a property for the entity that is being asked for. Therefore the focus denote a type or a property. Thus, the retrieved documents in which the system expects to find the answer may not always contain exactly the textual representation of the type or a property that was indicated by the focus. Focus is relevant for example in type coercing the candidate answers and in passage or evidence retrieval where the system tries to detect those types and properties in a textual content. Because of this, the increased performance of the system caused by making the focus text optional in the query is reasonable indeed.

In a couple of manually performed experiments, we partially validated the hypothesis that annotating the named entities in query formulation as a "MUST" increases the system's retrieval performance. In order to validate this, we arbitrarily choose 20 questions that the system cannot find a document that contains the expected answer using our current query formulation methodology. We manually annotated the named entities in the queries that are generated for these 20 questions as a must phrase, such as *"#band(#1(Ağrı Dağı))"*. After running with the new queries, we observe that the system can find relevant documents (i.e. the documents one of which contains the expected answer) for 16 questions. Therefore, we believe that a named entity recognition (NER) would greatly help in particular to the IR module.

7. CONCLUSIONS

In this study, we presented a novel combination of rule-based and statistical approaches to question analysis, employed in a closed-domain question answering system for an agglutinative language, such as Turkish. Our question analysis consists of focus extraction and question classification. For focus extraction, we have multiple rule-based experts for most frequent question types in Turkish. Additionally, we described a HMM-based novel sequence classification approach for focus extraction, along with combining the results of both rule-based and statistical models according to the individual confidence scores of each model. For question classification, we employed a rule-based classifier which uses pattern phrases unique to each class. We implemented baseline models for both problems, and have reported here the comparisons. We also designed and implemented a query formulation for unstructured search using Apache Lucene and Indri search engines to retrieve the relevant documents or passages from the given question using the results of the question analysis module. In addition to the methodologies offered, we also provide a set of manually annotated questions for both reproducibility and further research.

7.1. Future Work

HMM is very popular and powerful model, however, using it with only two states reduces its capabilities. Therefore we plan to change this model into Conditional Random Fields (CRF) that can learn directly the dependency tree structure and decide accordingly.

Additionally, a statistical approach for question classification should be developed, since the rule-based model will be manually improved further and further each time a new domain is added to the system (e.g. History, Chemistry etc.). For a coherent classification of the question, Support Vector Machines (SVM) using the features of the currently employed rule-based model is planned to be developed.

Increasing the capabilities of the IR module is planned to be developed further in future studies. Firstly, an approach for extracting structured information is planned. The module should extract small pieces of information from the dependency trees of the sentences in the textual resource base (currently having more than two million dependency parsed sentences). These pieces of information will constitute a factual information that can be used to produce aggregate statistics for relations between for example, subject and objects, objects and verbs etc. Additionally the previously learned factual information may help to produce candidate answers in an early stage in the system pipeline. For unstructured search, IR module is planned to be further improved by developing a consistent query term scoring mechanism. In order to achieve such a mechanism, a Named Entity Recognition (NER) component is planned to be developed. A NER component can detect the named entities in the given question, and the term scoring component can score the terms according to the named entities and the QClass information. For example, consider the following question.

"Nil Deltası hangi ülkenin sınırları içerisindedir?"

"Within the borders of which country lies the Nile River Delta?"

A NER component that can detect the "Nil Deltası" as a river delta may greatly help for example to identify it as a phrase and make possible to perform an informed search as to retrieve the documents related in general to deltas, which in turn greatly increases the probability that a suitable candidate answers to be determined.

It is also possible to improve the overall architecture of the system to parallelize the individual components to increase the time performance. Not only this will increase the system's modularity in a way that each parallelized component can be replaced, modified or improved individually, but this also makes possible to test and validate the results of each component on the fly within the compensated time frame.

REFERENCES

1. Ferrucci, D. A., "Introduction To "This Is Watson"", *IBM Journal of Research and Development*, Vol. 56, No 3.4, pp. 1-1, 2012.

2. Gupta, P. and V. Gupta, "A Survey of Text Question Answering Techniques", *International Journal of Computer Applications*, Vol. 53, No. 4, pp. 1-8, 2012.

3. Allam, A. M. N. and M. H. Haggag, "The Question Answering Systems: A Survey", *International Journal of Research and Reviews in Information Sciences* (IJRRIS), Vol. 2, No. 3, 2012.

4. Sahin, M., U. Sulubacak and G. Eryiğit, "Redefinition of Turkish Morphology Using Flag Diacritics", *In Proceedings of The Tenth Symposium on Natural Language Processing (SNLP-2013)*, Phuket, Thailand, October, 2013.

5. Eryiğit, G., "The Impact of Automatic Morphological Analysis & Disambiguation on Dependency Parsing of Turkish", *In LREC*, pp. 1960-1965, 2012.

6. Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov and E. Marsi, "MaltParser: A LanguageIndependent System for DataDiven Dependency Parsing", *Natural Language Engineering*, Vol. 13, No. 2, pp. 95-135, 2007.

7. Eryiğit, G., J. Nivre and K. Oflazer, "Dependency Parsing of Turkish", *Computational Linguistics*, Vol. 34, No. 3, pp. 357-389, 2008.

8. Dominguez-Sal, D. and M. Surdeanu, "A Machine Learning Approach for Factoid Question Answering", *Procesamiento del Lenguaje Natural*, No. 37, pp. 131-136, 2006.

9. Li, X. and D. Roth, "Learning Question Classifiers: The Role of Semantic Information", *Natural Language Engineering*, Vol. 12, No. 3, pp. 229-249, 2006.

10. Metzler, D. and W. B. Croft, "Analysis of Statistical Question Classification for FactBased Questions", *Information Retrieval*, Vol. 8, No. 3, pp. 481504, 2005.

11. Katz, B., "Annotating the World Wide Web using Natural Language", *In RIAO*, pp. 136-159, 1997.

12. Zheng, Z., "AnswerBus Question Answering System", *InProceedings of the second international conference on Human Language Technology Research*, pp. 399-404, Morgan Kaufmann Publishers Inc, 2002.

13. Brill, E., S. Dumais and M. Banko, "An Analysis of the AskMSR QuestionAnswering System", *In Proceedings of the ACL02 conference on Empirical methods in natural language processing*, Vol. 10, pp. 257264, *Association for Computational Linguistics*, 2002.

14. İlhan, S., N. Duru, Ş. Karagöz and M. Sağır, "Metin Madenciliği ile Soru Cevaplama Sistemi", 2008.

15. Er, N. P. and İ. Çiçekli, "A Factoid Question Answering System Using Answer Pattern Matching.", 2013.

16. Say, A. C., S. Demir, Ö. Çetinolu and F. Öün, "A Natural Language Processing Infrastructure for Turkish", *In Proceedings of the 20th international conference on Computational Linguistics*, pp. 1385. *Association for Computational Linguistics*, 2004.

17. Ferrucci, D., E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, ... and C. Welty, "Building Watson: An Overview of the DeepQA Project", *AI magazine*, Vol. 31, No. 3, pp. 59-79, Chicago, 2010.

18. Lally, A., J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, ... and J. ChuCarroll, "Question Analysis: How Watson Reads a Clue", *IBM Journal of Research and Development*, Vol. 56, No. 3.4, pp. 2-1, 2012.

19. Fan, J., A. Kalyanpur, D. C. Gondek and D. A. Ferrucci, "Automatic Knowledge Extraction from Documents", *IBM Journal of Research and Development*, Vol. 56, No. 3.4, pp. 5-1, 2012.

20. ChuCarroll, J., fffffJ. Fan, N. Schlaefer and W. Zadrozny, "Textual Resource Acquisition and Engineering", *IBM Journal of Research and Development* , Vol. 56, No. 3.4, pp. 4-1, 2012.

21. McCord, M. C., J. W. Murdock and B. K. Boguraev, "Deep Parsing in Watson", *IBM Journal of Research and Development*, Vol. 56, No. 3.4, 3-1, 2012.

22. Murdock, J. W., A. Kalyanpur, C. Welty, J. Fan, D. A. Ferrucci, D. C. Gondek, ... and H. Kanayama, "Typing Candidate Answers Using Type Coercion", *IBM Journal of Research and Development*, Vol. 56, No. 3.4, pp. 7-1, 2012.

23. Murdock, J. W., J. Fan, A. Lally, H. Shima and B. K. Boguraev, "Textual Evidence Gathering and Analysis", *IBM Journal of Research and Development*, Vol. 56, No. 3.4, pp. 8-1, 2012.

24. Gondek, D. C., A. Lally, A. Kalyanpur, J. W. Murdock, P. A. Duboue, L. Zhang, ... and C. Welty, "A Framework for Merging and Ranking of Answers in DeepQA", *IBM Journal of Research and Development*, Vol. 56, No. 3.4, pp. 14-1, 2012.

25. Delmonte, R. and E. Pianta, "Answering Why-Questions in Closed Domains From a Discourse Model", *In Proceedings of the 2008 Conference on Semantics in Text Processing*, pp. 103-114, *Association for Computational Linguistics*, 2008.

26. Bunescu, R. and Y. Huang, "Towards a General Model of Answer Typing: Question Focus Identification", *International Conference on Intelligent Text Processing and Computational Linguistics (CICLING)*, 2010.

27. Viterbi, A., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Transactions on Information Theory*, Vol. 13,

No. 2, 1967.

28. Wen, L., T. Amagasa and H. Kitagawa, "An Approach for XML Similarity Join Using Tree Serialization", *Database Systems for Advanced Applications, Lecture Notes in Computer Science*, Vol. 4947, pp. 562-570, 2008.

29. Munro, J. I. and V. Raman, "Succinct Representation of Balanced Parentheses and Static Trees", *SIAM J. Comput.*, Vol. 31, No. 3, pp. 762-776, 2002.

30. Benoit, D., D. E. Demaine, J. I. Munro and V. Raman, "Representing Trees of Higher Degree", *Algorithms and Data Structures, Lecture Notes in Computer Science*, Vol 1663, pp. 169-180, 1999.

31. Collins-Thompson, K., J. Callan, E. Terra and C. L. Clarke, "The Effect of Document Retrieval Quality on Factoid Question Answering Performance", *In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 574-575, ACM, 2004.

32. Srihari, R. and W. Li, *Information Extraction Supported Question Answering*, Cymfony Net Inc., Williamsville NY, 1999.

33. Hovy, E. H., U. Hermjakob and C. Y. Lin, "The Use of External Knowledge of Factoid QA", *In TREC*, Vol. 2001, pp. 644-52, 2001.

34. Suchanek, F. M., G. Kasneci and G. Weikum, "Yago: A Core of Semantic Knowledge", *In Proceedings of the 16th International Conference on World Wide Web*, pp. 697-706, ACM, 2007.

35. Miller, G. A., "WordNet: A Lexical Database for English", *Communications of the ACM*, Vol. 38, No. 11, pp. 39-41, 1995.

36. Tellex, S., B. Katz, J. Lin, A. Fernandes and G. Marton, "Quafntitative Evaluation of Passage Retrieval Algorithms for Question Answering", *In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development*

in Informaion Retrieval, pp. 41-47, ACM, 2003.

37. Stoyanchev, S., Y. C. Song and W. Lahti, "Exact Phrases in Information Retrieval for Question Answering", *In Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pp. 9-16, *Association for Computational Linguistics*, 2008.

38. Metzler, D. and W. B. Croft, "Combining the Language Model and Inference Network Approaches to Retrieval", Information Processing and Management, Vol. 40, No. 5, pp. 735-750, 2004.

39. Hatcher, E., O. Gospodnetic and M. McCandless, "Lucene in Action", 2004, https://softsearch2.googlecode.com/svn/trunk/Document/luceneinaction.pdf, [Accessed July 2014].

40. Chu-Carroll, J., J. Fan, B. K. Boguraev, D. Carmel, D. Sheinwald and C. Welty, "Finding Needles in the Haystack: Search and Candidate Generation", *IBM Journal of Research and Development*, Vol. 56, No. 3.4, pp. 6-1, 2012.