# TURKISH QUESTION ANSWERING
## *Question Answering for Distance Education Students*

Burcu Yurekli, Ahmet Arslan, Hakan G. Senel* and Ozgur Yilmazel
*Department of Computer Engineering, *Department of Electrical Engineering*
*Anadolu University, 2 Eylul Kampusu, Eskisehir, Turkey*

Keywords: Question answering, text extraction, indexing, searching, metadata generation, information retrieval.

Abstract: This paper reports on our progress towards building a Turkish Question Answering System to be used by distance education students of Anadolu University. We have converted 205 electronic books in PDF format to text, extracted metadata from these documents. Currently we provide search capabilities over these documents. Details on problems we faced, and solutions we came up with are provided. We also outline our plan for continuing and implementing a full Question Answering system for this domain. We also present ways to automatically evaluate our system.

## 1 INTRODUCTION

In the last decade online university education is becoming ever more popular, because it allows many people to use the new communication medium to get advanced degrees even if they can't attend to universities as full time students.

Anadolu University has been providing higher education opportunities through distance education since 1982 (with the establishment of Distance Education System) to those who could not otherwise continue their education. There are over one million students studying in this system in Turkey, and six European countries.

The learning materials are textbooks, TV programs, academic counseling services, videoconferences, computers and the Internet-based applications (http://www.anadolu.edu.tr). One of the internet based applications provided by the university is the e-book service, which allows students to access course books via internet. These course books are published in PDF format, and offered as links to full PDF files (~ 300 pages per book). In order to find the information, students either should look through these PDF documents or they should know exact place of the information, which means spending more time and effort.

The aim of this research is to build an intelligent question-answering system to improve distance education. The main objective of Question Answering (Q/A) is to identify the answer of a question in large collections of online documents (Harabagiu, Paşca, & Maiorano, 2000). By developing a Q/A system, we will enable students to get clear answers to their questions within context. Short answers within context will allow them to get to the relevant material faster, and ultimately improve their learning experience.

In this paper we present our progress in this research, discuss the challenges and our proposed solutions. Section 2 gives an overview of our system, provides details on different components and challenges. Section 3 gives details about the remaining work to complete our intelligent question answering system for the Anadolu University Distance Education students and section 4 summarizes our current progress and conclusions.

## 2 SYSTEM OVERVIEW

The prototype of the Anadolu University Distance Education Question-Answering System (AOF-QA) consists of five different processes: preprocessing, metadata generation, indexer, searcher, and QA module (see Figure 1). Each of the modules is described below.

### 2.1 Preprocessing

For automatic assignment of metadata, first the educational course books have been partitioned into



Yurekli B., Arslan A., G. Senel H. and Yilmazel O. (2008).
TURKISH QUESTION ANSWERING - Question Answering for Distance Education Students.
In *Proceedings of the Third International Conference on Software and Data Technologies - ISDM/ABF*, pages 320-323
DOI: 10.5220/0001900703200323

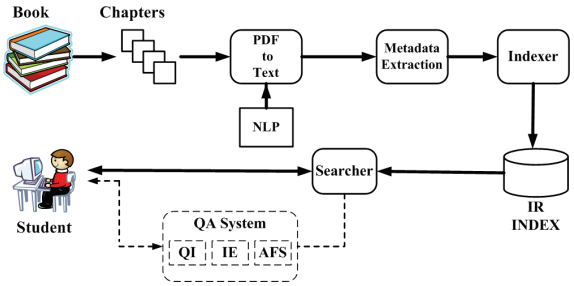


Figure 1: System overview. – implies future components.

chapters. Then, these chapters, which are in PDF format, have been converted to texts. Finally, from these texts the metadata have been generated.

### 2.1.1 Partition of Books into Chapters

The documents we have consist of course books in PDF format. Each of the course books has different number of chapters. To make metadata extraction available, we have divided the course books into smaller PDFs according to their chapters. The number of course books published in PDF format was about 230. Some of the course books were scanned through the hardcopy of the originals, so we had to eliminate them. Among the course books we have, 205 of them were partitioned into smaller PDFs. At last, we obtained 2654 PDFs, which is the number of the chapters of the 205 course books.

### 2.1.2 PDF to Text Conversion

In document processing, we have converted the PDF documents to texts to make them more suitable to allow processing. At this step, PDFBox (http://www.pdfbox.org), which is an open source Java PDF library to work with PDF documents, has been used.

While converting PDFs' into texts, we have faced some problems. Most of the PDF documents were legacy. Also, since they are written in Turkish, when we convert them to text, Turkish language specific characters like 'ı', 'İ', 'ğ', 'Ğ', 'ş', and 'Ş' were corrupted. Except 'ş' and 'Ş', the other corrupted Turkish characters have been corrected by replacement as each of them was referred by a non-Turkish single character. However we could not corrected 'ş' and 'Ş' characters by replacement. When these two characters are converted to text, 'ş' becomes 'fl' and 'Ş' becomes 'fi', and both 'fl' and 'fi' can take place in meaningful Turkish words. To overcome this issue, we thought that we need a spell checker, which will check if a word is correctly spelled or not. If there is a wrong spelled word, it will be changed by the correct one. As a spell checker, we used Zemberek, which is an open source Turkish NLP library (http://code.google.com/p/zemberek). Zemberek provides basic NLP operations such as spell checking, morphological parsing, stemming, word construction, word suggestion, converting words written only using ASCII characters (so called 'deasciifier') and extracting syllables.

Although Zemberek has overcomed many problems and been useful, in some cases it logically failed to do the right correction, because of the proper names and missspelled words. We created a correction map file, which contains a list of correct spellings of proper names and common words, to do the correction.

### 2.1.3 Metadata Extraction and Discourse Analysis

At this stage full-text of the chapters, which were obtained in the PDF to text conversion step, are converted into XML representations.

Instead of writing the full-text under a tag, we first extract the metadata such as author, summary, keywords and learning objectives so that we could display this information to the user in the result set. We followed similar research that has been done for (Yilmazel, Finneran, & Liddy, 2004) as it takes great amount of time and effort to create metadata of digital contents manually.

After the discourse analysis of the chapters, we had found that chapters were organized as chapter number, chapter title, introduction, text body, abstract, evaluation tests, and references. However, some of the chapters don't contain evaluation tests and references. Also, the introduction parts of the chapters show differences. Some of them may contain information like chapter author, aim, keywords and suggestions. We implemented a rule based extraction system to extract metadata of the chapter texts automatically.

We observed that our document collection could be separated into six categories according to the differences of the chapter full-texts. So, we designed a chapter parser which determines the category of the full-text. When a document is sent to this parser, it decides the category and extracts the metadata of the document.

Finally, we obtained the following metadata elements: Course No, Book Name, Book Author, Book ISBN, Chapter No, Chapter Title, Chapter Author, Chapter Begin, Foreword, Learning Objective, Keywords, Content, Suggestions,

Trouble, Introduction, and Text. While creating the XML documents, we have written these elements into corresponding tags.

At the forcoming processes of our research, the XML documents, obtained at this step, will be used as resources and our question answering system will depend on these documents.

We have built a windows application to check our system whether there exists any mistakes in the automatically extracted XML documents. By this application, we can see and compare the original PDF chapter and the extracted XML document. If there exists mistakes or unnecessary information like headers and footers, we can correct them manually in the XML document and create a new version.

## 2.2 Indexer

Many question answering systems use information retrieval to identify candidate documents that might contain the answer. To implement this candidate document selection phase, we used an open source information retrieval library called "Solr" (http://lucene.apache.org/solr). We modified the behaviour of Solr for Turkish language. In Solr, given documents are converted to token streams and each token is analyzed by different Analyzers or Filters before they get indexed in an inverted index for retrieval.

The following three filters were implemented:

i. **Turkish Lowercase Filter**

Lowercase filter that came with out-of-the-box settings of the Solr, lowercases letter 'I' to 'i'. However, in Turkish, lowercase of letter 'I' is 'ı'. Our Turkish Lowercase Filter lowercases terms correctly for the Turkish Alphabet.

ii. **Turkish Stem Filter**

Stemming is one of the well-known methods for improving Information Retrieval systems. Since Turkish is an agglutinative language, inflectional and derivational suffixes are used to create new words, stemming becomes more important in increasing recall and reducing the number of unique terms in the inverted index (Can et al., 2008). We implemented a stem filter, using Zemberek. Our stem filter identifies and removes inflectional suffixes from Turkish words.

iii. **Turkish Character Synonym Filter**

Turkish Language have (ç, ğ, ı, ö, ş, ü) six Turkish specific characters. At indexing time, if a token contains one of these six characters, this filter creates a new token replacing these characters with corresponding English characters and injects this new token to the token stream as a synonym of the original token. This is done because many Turkish computer users still choose to use US English Keyboards on their computers, that doesn't have the special characters, and transliterate Turkish words, by using the ASCII counter parts of the Turkish characters. (i.e. öğrenci vs. ogrenci).

Example: In our system if a user queries the word 'ogrenci' (not a valid Turkish word), he gets the results containing the word 'öğrenci' (student in English).

Table 1: The equivalence of Turkish specific characters in English.

| Turkish Letter | ç | ğ | ı | ö | ş | ü |
|---|---|---|---|---|---|---|
| English Letter | c | g | i | o | s | u |

## 2.3 Searcher

We developed a Web application that can return ranked lists of documents relevant to the given user query. Our results displayed the book name, chapter title, page number and highlighted text snippets from the document to the end user. Document metadata is also made available in the results screen by screen overlay. We thought that, this kind of output is suitable which is used by many search engines.

In our initial experiments we used book chapters as basic units of indexing. Doing chapter level indexing increased the highlighting of the results to about 7-8 seconds which is impractical for a live system.

We realized that, in order to obtain feasible response time, we had to decrease the highlighted field size. So, we decided to separate chapters into pages. We indexed our documents page by page. We used Solr's faceted search capability to group the result set according to chapters of books, and sort each group by page number. By this way, we obtained less than 500 milliseconds execution time for a query. This time is reasonable for the most live systems.

In addition to the highlighted snippets, a direct link to the page of the original PDF is also provided. Giving opportunity for accessing the original documents and to a specific page has many advantages like seeing figures, tables and visual components that cannot be converted to text.

## 2.4 QA Module – To be Implemented

This module will be used to answer a question. First, in the context of some ongoing interaction, our system will analyse the given question. Then by using the IR part of the research, we will retrieve the

relevant documents containing the answer of the question. Among these relevant documents, we will apply some answer finding algorithms, to extract one or more answers. Finally, in some appropriate form, we will present the answer to the user (Hirschman & Gaizauskas, 2001).

In our textbooks, at the end of each chapter there is an evaluation section. In this section there are multiple choice questions for the students to test themselves. Also, the correct answers of these questions are available. By using these data, we are going to construct a QA system evaluation data set, which will be used to test our QA system.

# 3 REMAINING WORK

Our research is going to expand from this information retrieval system to a Question-Answering System in multiple stages by implementing various modules needed.

First stage will be the implementation of the Question Interpreter for Turkish. This module will analyze the incoming question to produce a logical representation for the question that can be converted to a Boolean query for information retrieval. We envision this module to have synonym expansion, spelling correction, and other further processing needed for handling questions.

In order to answer many factoid and comparison questions we would need to have structured information extracted from the documents. We will implement an Information Extraction module to mark texts within these books with structured information.

Final module in our QA system is actually multiple modules, Answer Finding Strategy (AFS). Each AFS will implement a different answer finding strategy to search for an answer in the documents and consolidate these for displaying to the user.

We will tune and evaluate our QA system from the Question Repositories of AOF exams. This is a unique resource and is not available to many QA researchers. Anadolu University Distance Education System requires every question asked at an exam be linked to a section or a paragraph in the course books. By using this information we could easily generate an automated QA evaluation set for a large number of questions. In educational settings students evaluate a question answering system in many other dimensions only one of which is the relevancy of the results (Liddy, Diekema, & Yilmazel, 2004). We will conduct user studies to measure user satisfaction in various dimensions.

# 4 CONCLUSIONS

During the process of developing an intelligent question-answering system for students of Anadolu University Distance Education System, so far we have completed the preprocessing and metadata extraction of the document collection. We can now support faceted searches over the full collection of books. Our search technology is aware of the properties of the Turkish language. Our system is now ready for beta testing, which we plan on deploying very soon. With over one million registered students, we hope to get valuable feedback from the users on the retrieval quality.

The NLP processes for non-English languages, like Turkish, are hard, due to limited resources and character encoding problems.

We will continue to publish our progress as we go forward with our Question Answering system. This will be the first large scale question answering system for Turkish, to best of our knowledge, and we hope that it will be helpful for our students.

# ACKNOWLEDGEMENTS

This research has been supported by Anadolu University as it gives permission to work on Distance Education System's course materials.

# REFERENCES

Can, F., Koçberber, S., Balçık, E., Kaynak, C., Öcalan, H. Ç., & Vursavaş, O. M. (2008). IR on Turkish Texts. Journal of the American Society for Information Science and Technology.

Harabagiu, S. M., Paşca, M. A., & Maiorano, S. J. (2000). Experiments with Open-Domain Textual QA. Proceedings of the 18th conference on Computational linguistics - Volume 1.

Hirschman, L., & Gaizauskas, R. (2001). NL QA: The View from Here. Nat. Lang. Eng., 7(4), 275-300.

Liddy, E. D., Diekema, A. R., & Yilmazel, O. (2004). Context-Based QA Evaluation, Sheffield, UK.

Yilmazel, O., Finneran, C. M., & Liddy, E. D. (2004). Metaextract: An NLP System to Automatically Assign Metadata. Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries.