A prototype framework for a Bangla question answering system using translation based on transliteration and table look-up as an interface for the medical domain

Nafid Haque
M.Sc HLST
April 2010

University of Malta

Department of Computer Science and Artificial Intelligence

Masters Thesis

Nafid Haque

A prototype framework for a Bangla question answering system using translation based on transliteration and table look-up as an interface for the medical domain

Supervisor(s) :
Mike Rosner, University of Malta
Gertjan Van Noord, University of Groningen

Programme: M.Sc HLST
Specialization: Erasmus Mundus Masters Program in Language and Communication Technologies (LCT)

April 2010

University of Groningen

Faculty of Arts

Masters Thesis

Nafid Haque

A prototype framework for a Bangla question answering system using translation based on transliteration and table look-up as an interface for the medical domain

Supervisor(s) :
Gertjan Van Noord, University of Groningen
Mike Rosner, University of Malta

Programme: Research Master Linguistics
Specialization: Erasmus Mundus Masters Program in Language and Communication Technologies (LCT)

April 2010

University of Groningen

Faculty of Arts

Masters Thesis

Nafid Haque

A prototype framework for a Bangla question answering system using translation based on transliteration and table look-up as an interface for the medical domain

Supervisor(s) :
Gertjan Van Noord, University of Groningen
Mike Rosner, University of Malta

Programme: Research Master Linguistics
Specialization: Erasmus Mundus Masters Program in Language and Communication Technologies (LCT)

April 2010

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own work, of work of another person, whether or not such work has been published" (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

I / ~~We*~~, the undersigned, declare that the [~~assignment /~~ ~~Assigned Practical Task report~~ / Final Year Project report] submitted is my / ~~our*~~ work, except where acknowledged and referenced.

I / ~~We*~~ understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N.B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

NAFID HAQUE
Student Name

Nafid Haque
Signature

Student Name

Signature

Student Name

Signature

Student Name

Signature

CSA5310
Course Code

MSc HLST Dissertation : A prototype framework for Bangla Question Answering system using translation based on transliteration and table-lookup as an interface for the Medical Domain
Title of work submitted

2/4/2010.
Date

To the martyrs of The Language Movement of 1952…thank you

Acknowledgement

I would like to thank my supervisors and specially Mike Rosner for holding my spirits during my bad times.

Special thanks to the organizers of the European Masters in Language and Communication Technology (EM-LCT) to present such a nice program. This program has taught me many things and I am grateful to each and every individual professors, lecturers and local co-ordinators of this program who has helped me somehow in the past two years.

My family, whom I cannot thank enough for all their inspiration and support throughout my life. Without their inspiration and support nothing would have been possible.

I am also very thankful to my employer and all my colleagues who understood my situation and gave me all the time and support needed to complete this work.

Lastly I would like to thank all the friends and colleagues of the EM-LCT program for helping me and inspiring me now and then.

Thank You.

Nafid H.

Abstract

Question Answering (QA) Systems allow the user to ask questions in a natural language and obtain an exact answer. Through this thesis work, we tried to learn the important issues in the field of Question Answering (QA) systems. We peeked into the internals of many established QA systems. We explored the capabilities of each of them and the reasons that make them good at their task. Then we looked into the details of cross-language QA task. We learned that most of such systems employ some form of machine translation engines. We aimed to have a complete cross-language QA system for Bangla. The language Bangla is among one of the most widely spoken languages of the world but is still in its early stages of research regarding language processing resources and tools. Thus for the cross-language QA task we did not have access to translation engine which was very essential. So we narrowed down our aim and finally proposed an innovative concept of translation based on transliteration and a table look-up approach as an interface for a cross-language QA task where one of the languages involved is at a disadvantage in terms of digital language resources and tools. The proposed concept is implemented in a form of a prototype framework for a very controlled cross-language QA scenario. We do not claim that our proposed approach is a complete approach for a Bangla QA task but we did achieve promising results that can help in Bangla QA task until mature Bangla language processing tools become available.

Table of Contents

List of Tables

List of Figures

1 Introduction

Question Answering (QA) systems go beyond the usual Information Retrieval (IR) systems which underly popular Internet search engines. QA systems have the aim of responding to natural language questions whereas IR systems take up keywords from users and deploy some intelligent search mechanisms on a document collection to get back to the user with a ranked list of documents rather than an exact answer. However, the user still has to go through the documents to find out the exact answer of his or her query. This process of going through the documents to find an answer is undesirable and QA systems, by contrast, are expected to eliminate this process by giving an exact answer to a question. Thus the aim of a QA system is to localize the exact answer to a question from a structured or a non-structured collection of texts.

Asking questions in natural language and obtaining exact answers make QA systems of paramount importance to Information Retrieval [Laurent et al. 2006]. Previously, given an information need of a user, systems retrieved information from a text collection by retrieving full-length documents; however, in recent times the focus of these systems has moved to giving the specific information rather than a bibliographic-like information.

The design of a standard QA system assumes that the language in which the question is asked and the text collection available to be processed are all in the same language. However, there might be a need for cross-lingual QA system which take in questions in one language and searches through a document collection in a different language to get to the answer.

In this thesis work, we discuss the issues to look at to build a cross-language QA system and finally present a model framework for such a cross-language QA system, where one of the languages (Bangla) has very limited computational resources to have a complete QA system of its own. We start by giving an introduction to general QA systems and their associated components and slowly build the discussion towards a cross-language QA task. Then we introduce our research aim and finally present what we have achieved through this work.

1.1 Generic Question Answering (QA) Systems

The basic architecture of a QA system is dependent mostly upon the anticipated user of the system, the type of questions to be handled by the system, the type of expected answers and the format in which the available information is stored [Monz 2003]. The possibility of the information to be available in different formats makes the entire design of a QA system a bit more complex, not to mention the final performance of the system. It is possible that the QA system tries to answer a question by accessing a structured information source such as a database or an unstructured information source such as plain text documents. It is also possible to have a hybrid system that can handle both structured and unstructured data.

Those systems that have a structured knowledge-base mostly exploit that structure to produce a match between the question and an answer [Monz 2003]. This type of system is relatively easy to build compared to the ones having an unstructured knowledge-base. Unstructured information is usually in plain-text format such as articles from newspaper, manuals, encyclopedias etc. QA systems having an unstructured knowledge-base try to find a match between the text units in the collection and the question itself to get to an answer. Thus the text units in the collection need to be descriptive enough about their own structure as well as the content itself for the system to make some intelligent use of them to reach to an answer.

[Pasca 2003] states that a QA task can be decomposed into three main subproblems. The subproblems are:

1. Question Processing
2. Document Processing
3. Answer Processing

The *question processing* stage is responsible for taking a question in a natural language and producing some kind of intelligent representation of the raw question string so that it becomes more useful for finding answers. The *document processing* stage is used to reduce the search space of the document collection where the answer to the question can be expected. This stage is basically a complete Information Retrieval system where the idea is to take in some keywords and produce a ranked list of documents related to those keywords. The final stage of a QA system is the *answer processing* stage where the system does some intelligent matching with the output of the previous

two stages to produce an answer to the given question. Any QA system should have these three basic components and may have a number of other components to make the system more useful and robust. The general architecture of a QA system can be modeled like the diagram shown below.

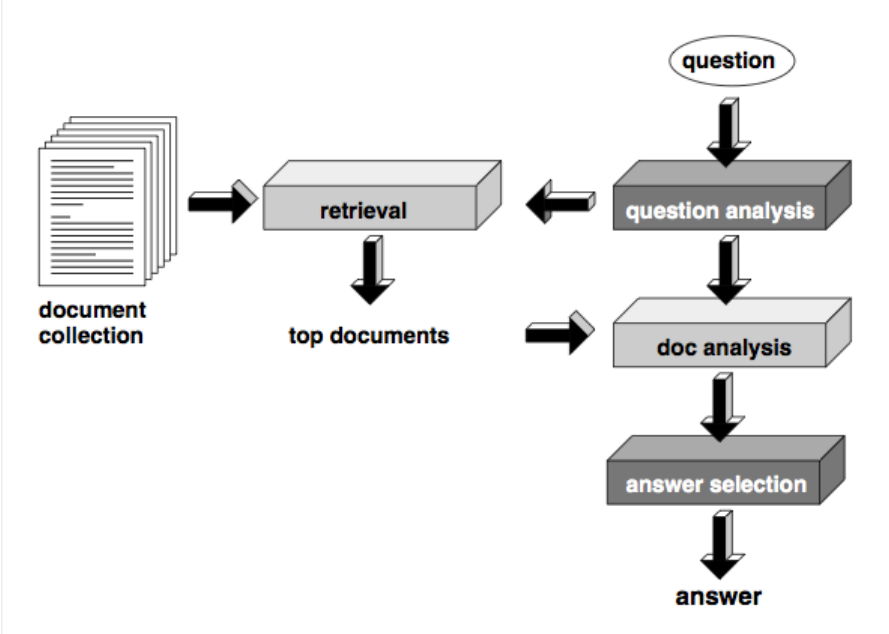

Fig 1: Components of a generic QA system [Monz 2003]

There are many other components such as a parser, part-of-speech tagger, stemmer, named entity recognizer and automatic machine translation engine which can be included in the skeleton system to improve the performance and solve other complex issues in QA. All these additional components may be merged between the 3 main components or may stay as individual black boxes to assist in the overall task of question answering. Some of these important additional components will be discussed in detail after some detailed discussions on the 3 major components of a QA system.

1.1.1 Question Processing

The main function of this component is to analyze the question taken from the user. Questions can be of different types and classifications thus this component is responsible to identify in which class the question falls. The question type derived here can be used for answer extraction and answer filtering to improve the accuracy of the overall QA task. The question processing component also needs to determine the expected answer type (EAT). A QA system can be made to work with only a certain class of questions or it may be built in such a way as to entertain a wide variety of classes. Further the system could entertain questions from a certain topic, making the entire system closed to a single domain, or it could entertain varieties of topic, making the system respond to an open-

domain. Some of the possible question classifications are shown in Table 1.

Question Type	Description
Agent	Name or description of an animate entity causing an action *Who won the Oscar for best actor in 1970?*
AKA	Alternative name for some entity *What is the fear of lightning called?*
Capital	Capital of a state or country *What is the capital of Kentucky?*
Date	Date of an event *When did the story of Romeo and Juliet take place?*
Date-birth	Date of birth of some person *When was King Louis XIV born?*
Date-death	Date of death of some person *When did Einstein die?*
Expand-abbr	The full meaning of an abbreviation *What does NASDAQ stand for?*
Location	Location of some entity or event *Where did Golda Meir grow up?*

Table 1: Possible question types [Monz 2003]

Question classification can be done in roughly two major ways, namely, a rule-based approach or a statistical approach [Day et al. 2005]. There are many ways to identify in which class a question belongs as stated in Monz [2003]. The most common and simple way is to look for patterns in the incoming question which fall under the rule-based approach category. The task of pattern matching can be achieved by handcrafting different regular expressions as can be seen in Table 2.

Question class	Example patterns
agent	/[Ww]ho /,/ by whom[\.\?]/
aka	/[Ww]hat(i|\')s (another|different) name /
capital	/[Ww]hat is the capital /,/[Ww]hat is .+\'s capital/
date	/[Ww]hen /,/[Ww](hat|hich) year /
date-birth	/[Ww]hen .* born/,/[Ww](hat|hich) year .* born/

Table 2: Sample patterns to classify question [Monz 2003]

For a statistical approach towards question classification, expert knowledge is used to prepare a sufficiently large collection of data which in this case would be a collection of question and answer pairs. A model is trained to automatically capture all the useful patterns for question classification. The statistical approach to question classification can be further enhanced with different machine

learning models to improve the performance of the question processing component as well as the overall QA task. Zaanen et al. [2005] proposed a combination of the machine learning and pattern matching approach to question classification. They use an Alignment-Based Learning classifier to learn structure from plain text sentences. They train the model with pairs consisting of regular expressions found and the corresponding expected answer type (EAT). Several questions can match a single regular expression and thus have more than one EATs. During the classification task all the regular expressions are tried and the EAT with the highest frequency is chosen. They also propose an approach with a Trie[1] classifier to determine the type of question. Their system learns from questions inserted in a trie structure that contains the token, the EAT and the frequency information (the number of questions that use that single path in the trie). During the classification task the trie is traversed and if a new question is a prefix of a training question then the node at the end of the traversal path indicates the EAT of the question, otherwise a lookahead approach is used on the sub-tries until all the tokens are consumed and a path with the highest frequency is reached. Day et al. [2005] uses INFOMAP[2] and Support Vector Machines (SVM)[3] to classify Chinese questions. They develop a hierarchical two-layer taxonomy comprising of the question type or the EAT by analyzing the TREC[4] question corpus. Then they use INFOMAP to identify the category of the Chinese questions. If the knowledge-based approach fails to identify a category for the question then an SVM model is used as a fallback. The SVM model uses syntactic features like part-of-speech (POS) and other models like bag-of-words[5] to classify the question. It further uses semantic features from another ontology database called HowNet 2000[6] to classify the question. Tomas et al. [2009] proposed a semi-supervised approach called the semantic kernels for question classification. In their approach they put the input data, which is the question, in a suitable feature space and then use a

1 Tries or digital trees are both an abstract structure and a data structure that can be superimposed on a set of strings over some fixed alphabet. As an abstract structure they are based on splitting according to letters encountered in strings: if S is a set of strings and $A = \{a_j\}_{j=1}^{r}$ is the alphabet, then the trie associated to S is defined recursively by the rule: trie (S) = {trie(S/a_1)...trie(S/a_r)}, where S/a means the subset of S consisting of strings that start with a_i stripped of their initial letter a, recursion is halted as soon as S contains less than 2 elements. The advantage of the trie is that it only maintains the minimal prefix set of characters that is necessary to distinguish all the elements of S. The trie(S) supports the search for any string w in the set S by following an access path dictated by the successive letters of w. [Clement at al. 1997]

2 A knowledge representation and inference engine. It is used to facilitate knowledge sharing by different application systems. When a QA system receives a query, it extracts the corresponding events or scripts based on the ontology in INFOMAP. [Hsu et al. 2001]

3 SVM is a supervised learning algorithm to classify elements.

4 The Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense, was started in 1992 to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. Since TREC-8 (1999) TREC introduced a question answering track.

5 Bag-of-words model represents a piece of text as an unordered collection of words without taking into consideration the grammar and word ordering of the piece of text.

6 HowNet is an online common-sense knowledge base unveiling inter-conceptual relations and inter-attribute relations of concepts as connoting in lexicons of the Chinese and their English equivalent. [Dong et al. 1999]

kernel function to discover any nonlinear pattern in the input space. The kernel function gives a similarity measure between the input data that depends exclusively on the specific data type and domain. Bouma et al. [2006a] uses dependency relations of the given question to determine the question type. That approach is discussed in detail in chapter 2.

As can be seen that whatever the approach is to analyze the question (rule-based or statistical), some kind of morpho-syntactic analysis and processing is required on the question itself such as finding out the part-of-speech, the root form (stemming) and the cardinality. Once these are found, it is the time to formulate a query that is to be used by the next component, which is the document processing unit. Formulating a query is dependent upon the structure of the document processing unit.

The way queries are formulated has a strong impact on retrieval effectiveness even if query formulation is just based on term selection without expanding the queries with semantically related terms [Monz 2007]. The most common and simple way is to identify keyword(s) from the question, finding the morphological root forms of the keyword(s), using some boolean operators with them and producing a query.

e.g Q: What is the abbreviation for United Nations?

A system can exclude the question terms (like *What, When, Who*) and just include the other terms as a boolean query.

e.g abbreviation AND United AND Nations.

However, this simple approach can steer the retrieval process in a wrong direction because most documents that contain an answer to the question asked might contain sentences like "...United Nations (UN)...", thus not using all the query terms. Both simple boolean conjunction of Bag-of-Words as well as Vector-space retrieval[7] will prefer documents containing all the terms over documents that do not contain a term. Stemming a query term can help in overcoming vocabulary mismatches. In case there are quotes in the question, then the entire quotation is treated as a phrase and constituent words are not used as query terms. Once a question is POS tagged, a phrase level

7 Vector space model is used to represent text documents as an algebraic model. The model creates a space in which both documents and queries are represented by vectors. For a fixed collection of documents, an *m* dimensional vector is generated for each document and each query form sets of terms with associated weights, where *m* is the number of unique terms in the documents collection. A vector similarity function (like inner product) can be used to compute the similarity between a document and a query. [http://www2002.org/CDROM/refereed/643/node5.html]

analysis can be done to identify query terms. Terra et al. [2005] looks for some patterns to identify noun phrases such as: 1) adjective followed by noun; 2) a non-proper noun followed by any noun; 3) foreign word followed by any noun; 4) any noun followed by a foreign word; 5) proper-noun followed by proper noun; and 5) numeral followed by any noun. The intuition behind is that the entire noun phrase conveys more information rather than the individual terms. The noun phrase can be put inside quotations for better results. More details about document retrieval can be found in the document processing section.

Monz [2007] proposed a machine learning approach for query term weighting to improve the retrieval engine. He analyzed TREC (TREC 9,10,11) datasets to compute the optimal term selection for each question. For a given question *q* having *T* terms, all the possible subsets of *T* are considered and evaluated. That is, the set of term selection variants *(tsv)* is defined as *tsv(q) = POW(T) – {0}*. Monz [2007] determined the query variant with the highest average precision for each question in the 3 datasets. He suggests that if a term occurs in query variants that have a high average precision it should have a high weight and a term that occurs in query variants that have a low average precision then it should receive low weight. Thus, the weight of a query term depends on two factors: its presence weight $w_+(t)$ and its absence weight $w_-(t)$. Those weights are normalized and combined into a single weight by subtracting the absence weight from the presence weight which is called the gain of term *t* ($gain(t)=w_+(t)-w_-(t)$). If a term *t* has a positive gain, then it should be included, otherwise not. This approach to computing the term weights is solely based on the distribution of the terms over the query variants. This leads to problems such as terms having a high gain for one query and low gain for another. Also, this computation is not possible for terms missing in the training data. Thus, Monz [2007] introduces a list of features to further enhance the computation of the term weights. An extract of the features is shown Table 3.

Feature	Value	Feature	Value
part-of-speech	Penn Treebank part-of-speech tag	location	Whether the word is part of a location name
question focus	Whether the word is part of the question focus	abbreviation	Whether the word is an abbreviation
superlative	Whether the question contains a superlative adjective	upper case	Whether the word starts with an uppercase letter
question class	A fixed list of question classes	classif. word	Whether the word was used to classify the question
multpl. occurr.	Whether the word occurs more than one in the question	person name	What part of a person's name is the word, if applicable
quoted	Whether the word occurs between quotation marks	honorific	Whether the word is a honorific term (e.g., Dr.)
modified noun	Whether the word is a noun that is preceded (modified) by another noun	no. edges	The number of edges pointing to a word in the dependency parse graph of the question
term ratio	$1/m$, where m is the number of unique terms in the question	hypernym	Whether the word is a hypernym of another question word
no. leaves	The number of hyponyms in WordNet that do not have any further hyponyms	relative idf	The relative idf compared to the other words in the question

Table 3: List of features for question words. [Monz 2007]

Monz [2007] used the M5'[8] algorithm to assign weights to query terms, where the input to the learning algorithm is the set of feature vectors and the term gain is calculated from the training data. They incorporated the learned weights in to an IR engine and observed modest improvements, with some significant improvements in some cases.

Some query terms might need to be expanded to increase the probability of finding documents containing an answer. A semantic knowledge base like WordNet[9] can help to identify synonyms.

e.g Q: Where is Big Ben located?

The term "located" might not be enough (because of the system design) to search the text collection as there might be sentences like "You can find Big Ben in London." or "Big Ben is situated in London." Both the sentences are possible answers to the question but if the system is not aware that "locate" is semantically equivalent to "find" or "situate" those sentences might be discarded though

8 M5' is a reconstruction of the Quinlan's M5 algorithm. The M5 algorithm builds model trees combining conventional decision tree learning with the possibility of linear regression models at the leaves of the tree. M5 is suited for learning query term weights as it allows to consider dependencies between features [Monz 2007].

9 WordNet is a large lexical database of English available freely and publicly. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet's structure makes it a useful tool for computational linguistics and natural language processing. [http://wordnet.princeton.edu/]

being favorable candidates as an answer.

1.1.2 Document Processing

The document processing unit is just a regular IR engine which takes in a query and identifies some documents from its collection that are likely to contain an answer. Choosing the most appropriate query terms are very essential to get the most relevant documents. This unit is not responsible for finding an actual answer to the question. However, the performance of this component is critical to the overall performance of the entire QA system. The collection of text and its format is also an important factor. The text collection could be a closed corpus with a limited amount of text or it can be a dynamic corpus which changes over time such as the Internet. A limited document set is much easier to handle as many things can be hard-coded to get a better performance. Bouma et al. [2006a] talk about *Linguistically Informed Information Retrieval*. Bouma et al. [2006a] perform a full syntactic analysis of their text collection over different linguistic dimensions such as POS tags, NE tags and dependency relations. Using these linguistic features they index their data so as to improve the performance of their retrieval task. A more detailed description of Linguistically Informed Information Retrieval approaches can be found in the chapter 2.

In case the text collection is open and dynamic, the IR engine should be aware of that fact and index the new information in a timely fashion so that the new information is searchable. In such cases it is more likely that more relevant information can be obtained but this comes with some unavoidable and additional overheads. Documents over the Internet can be structured in many different ways and thus the system has to take care of such issues. And as these documents/text collection can change frequently, it is not possible to do some kind of preprocessing from earlier as suggested by Bouma et al. [2006a], instead everything needs to be done at runtime.

The IR engine can retrieve documents with only one of the keywords being present in the document or it can retrieve documents with all of the keywords being present in each of the documents. ZPRISE[10] IR is one such engine which does not retrieve documents having all the keywords. It uses a cosine vector space model where extraction of documents is based on a similarity measure between the document and the query. This allows extraction of documents when only one of the keywords is present. Furthermore all retrieved documents may not need to contain the same

10 ZPRISE is a public domain IR engine based on the NIST PRISE system that treats documents and queries as lists of words and responds to a query with a list of documents ranked in order of their statistical similarity to the query. [http://www-nlpir.nist.gov/works/papers/zp2/intro.html]

keywords. Moldovan et al. [2000]'s LASSO system is based on the principle that documents are retrieved only when all of the keywords are present in the document. It is implemented using Boolean indexing[11] as they claim that Boolean indexing increases the recall at the expense of precision.

This unit can be tailored further to behave like an intelligent IR engine that produces passages that may contain an answer rather than listing an entire document that may contain the answer. In passage-based retrieval, documents are divided into several passages and the size of the passages could vary depending upon the implementation. Monz [2003] states passage-based retrieval proves to be very useful in the QA task as information sought in a QA system tend to be found in a sentence or two. Thus for a document *D* there would be several passages like P_1, P_2, P_3 ... P_n. And for the query *Q* the relevant passage is P_2. So instead of calculating some kind of similarity measure between *D* and *Q*, similarity measures between P_i and *Q* are checked for *i* between *1* to *n*. The passage having a greater similarity value with the query is more likely to contain the answer or help in generating the answer. Moldovan et al. [2000] uses a *PARAGRAPH n* operator to filter out paragraphs. The *PARAGRAPH n* operator searches like an AND operator for the words in the query with the constraint that the words belong only to some *n* consecutive paragraphs, where *n* is a controllable positive integer. The parameter *n* selects the number of paragraphs, thus controlling the size of the text retrieved from a document considered relevant.

Moldovan et al. [2000], after obtaining a list of paragraphs in the LASSO system, performs a paragraph ordering based on radix[12] sort. It takes into account paragraph-windows[13] based on three different scores that are 1) the largest Same_word_sequence-score, 2) the largest Distance-score and

11 In Boolean indexing, documents are represented as words with their position information. Queries are expressions composed of words and connectives such as “and”, “or”, “not” and proximity operators such as “within *k* words of”. The answer to the query is the set of all the documents that satisfy a query. [Harabagiu et al. 1999]

12 Radix sort is a linear sorting algorithm that functions by sorting the input numbers/words by each digit/character for each digit/character in the number/word.

13 Paragraph-windows are determined by the need to consider separately each match of the same keyword in the same paragraph. For a set of keywords $\{k_1, k_2, k_3, k_4\}$ suppose k_1 and k_2 are each matched twice in a paragraph, k_3 is matched only once and k_4 is not matched, then there will be 4 windows defined as $[k_1\text{-match1}, k_2\text{-match1}, k_3]$, $[k_1\text{-match2}, k_2\text{-match1}, k_3]$, $[k_1\text{-match1}, k_2\text{-match2}, k_3]$, and $[k_1\text{-match2}, k_2\text{-match2}, k_3]$. Each of these windows consist of all the text between the lowest positioned keyword in the window and the highest position keyword in the window. For each such windows 3 scores are calculated [Moldovan et al. 2000].

1. *Same_word_sequence-score* – which is the number of words from the question that are recognized in the same sequence in the current paragraph-window.
2. *Distance-score* – which is the number of words that separate the most distant key-words in the window.
3. *Missing_keywords-score* – which is the number of unmatched keywords.

3) the smallest Missing_keyword-score. Finally, a radix sorting is done across all the window scores for all paragraphs.

1.1.3 Answer Processing

The final component is responsible for analyzing the documents or passages returned by the previous unit and finally identify possibly a single answer (could be a ranked list of answers, too) to the question. The passages or documents retrieved are based on the query terms used in the first component (IR engine) and those query terms are identified while processing the question itself in the Question Processing stage. Moldovan et al. [2000] states that recognition of the answer type is crucial to the identification of the answer. Thus we have already noticed that during the Question Processing stage an expected answer type (EAT) is also formulated by most of the systems. From Table 1 we have seen that each question is classified using possible question types. Those question types give a hint of the possible answer that is expected for that question.

e.g. What is the capital of Bangladesh?

The above question can be classified as a location type question with an additional constraint/information that the location must be a capital. Thus let the question type for the above question be LOC-CAPITAL. Now if the system manages to identify a sentence like

e.g. The capital of Bangladesh is Dhaka.

having a term "Dhaka" labelled as LOC-CAPITAL and the sentence happens to contain the word "Bangladesh" then that sentence can be considered a strong candidate for an answer. So we notice that identifying the question type directly or indirectly helps in answer processing. The answer extraction and processing part can employ many creative ways to improve the overall performance. Jijkoun et al. [2007] implements 3 different approaches to answer extraction in their Quartz QA system. In the first approach a table look-up method is searched for answers. The table is built offline using predefined rules to extract specialized knowledge. The rules basically take advantage of EATs such as location, dates etc. that are easily identifiable to build the offline table. At runtime a match is looked for between the question and the entries in the table. Bouma et al. [2006a] also implement such a table look-up method, which is discussed in the next chapter. The second approach in Jijkoun et al. [2007] looks for answers by searching for the most frequent word n-grams in the list of passages retrieved from the document processing stage. The third approach is similar to the second approach but instead of searching for answers from the passages obtained from the corpus, it tries to retrieve answers from the text snippets returned by the online web search engine

Google[14]. Monz [2003] talks about syntactic structure matching, pattern matching, lexical chaining or linear proximity methods to find a possible link between the given question, the possible answer pattern and the retrieved passages from the previous components to identify the likely answer to the question. This unit is highly dependent on the creativity of the system designer to find the precise answer to the question. The document collection may have an exact answer to the question or might contain facts from which the exact answer is to be inferred. The performance of the overall system is also dependent on the fact that how closely the question is matched with a passage that may contain the answer and finally how that answer is extracted or generated. Bouma et al. [2006a] states different ranking methods to rank the answers when there is more than one answer.

1.1.4 Additional Important Components

Here we introduce some special components that may work as add-ons to the skeleton architecture of a QA system. Not all these components need to be present in a basic QA system; however, several research works have shown how such add-ons influence in the overall performance of a QA task. This section give general detail of such add-ons. Their respective uses are discussed in detail in relevant sections of this thesis.

A parser is a program which in its simplest form checks for the grammatical consistency of a sentence and builds a hierarchical data structure by following a set of rules. In natural language processing, parsing is a method to perform some form of syntactic analysis of a sentence. The end result of parsing is one or more parse trees giving detailed structural information about the syntax of a sentence. This structural information comes in useful for many natural language processing tasks such as information extraction (IE), sentence generation and especially question answering. We will discuss here how and which parsing technologies are useful in a QA task.

A syntactic parser will produce a syntactic parse tree (Fig. 2) which gives information about the syntax of a sentence. The tree structure will help in identifying the constituents, such as noun phrases and verb phrases, but does not give further internal information such as dependencies between the tokens/constituents etc.

14 Google is a popular search engine owned by Google Inc.. It scans web pages to find instances of the keywords entered as query terms.

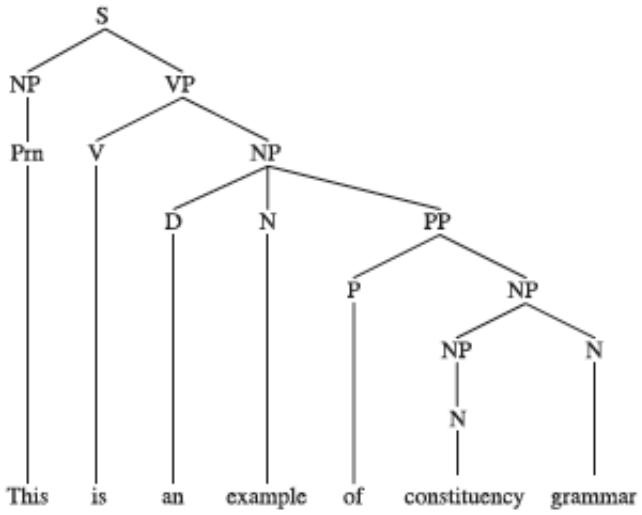

Fig 2: Syntactic parse tree (Constituency tree) [Covington 2000]

A dependency parser on the other hand produces a dependency tree (Fig. 3) which looks into the concept of a word-to-word link to identify any semantic relations between words. Thus whenever two words are connected by a dependency relation, one of them is the head playing the larger role in determining the behavior of the pair and the other is its dependent, which acts as a modifier, object or complement to the head. The dependent presupposes the presence of the head and the head requires the presence of the dependent too [Covington 2000].

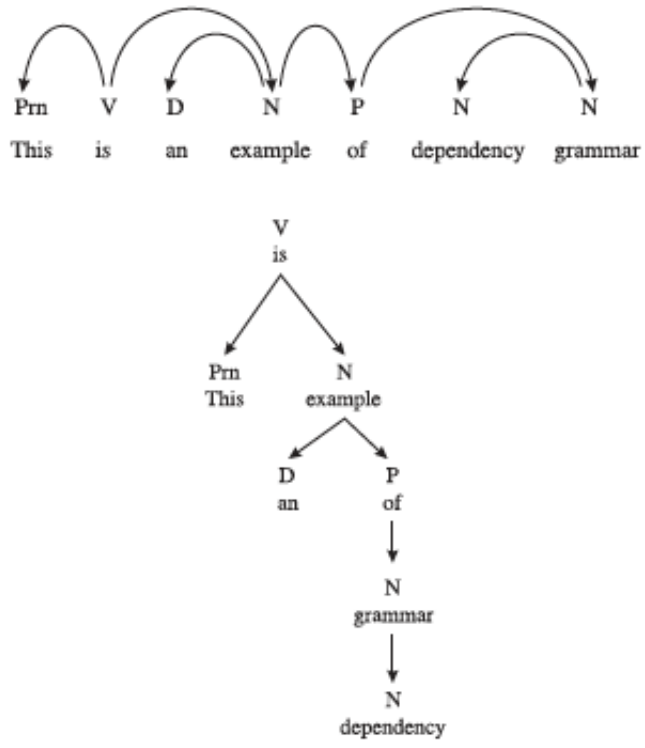

Fig 3: Dependency parse tree [Covington 2000]

A dependency parse tree may not preserve the word ordering of sentences but gives more information about which word is dependent on which one. A dependent that precedes its head is called a pre-dependent, and post-dependent follows the head. Additional structural information from the parse trees can be utilized in many ways within a QA task. A shallow parsing[15] of the question can help in identifying a clause or a phrase and thus could ultimately help in choosing the query terms for the retrieval component. A dependency relation can further help in identifying the head and thus helping in matching a question with an answer.

A POS tagger marks up the words in a sentence to a particular part-of-speech based on its definition as well as the context. This helps the parser to produce the structural information.

A Named Entity (NE) tagger is similar to a POS tagger but only identifies and tags some predefined categories such as names of persons, organizations, locations etc. An NE tagger is very important tool for a QA task. Sometimes some adjectives maybe part of a proper/common noun and thus if the adjective is considered literally then it might lead to something unexpected.

e.g. Which city is known as the Big Apple?

For the above question, though "big" itself is an adjective, in the example it is actually part of the named entity "Big Apple". Thus the QA system needs a mechanism to tag "Big Apple" as a named entity and search for "Big Apple" as a single term rather than considering them as individual terms and omitting one or the other while formulating the query. Further, when an expected answer type (EAT) is formulated before an actual answer is obtained the EAT can point to a specific NE class making the answer extraction task easier.

e.g. Where is the river Nile?

For this question a possible EAT is a location. Thus the retrieved passages/documents should contain some entities marked as locations.

A stemmer extracts the morphological root form of a word, e.g. "Dietary", "dietician" are reduced to its root form "diet". Stemming helps in formulating the query term for the document retrieval component. Monz [2003] states that some QA systems do not use stemming to avoid compromising

15 Shallow parsing (chunking or light parsing) is an analysis of a sentence which identifies its constituents without giving much information about the sentence's internal structure.

with early precision; however, some hybrid approaches consider both the root and actual one to improve the document similarity during the document retrieval task.

A machine translation engine takes in a text in one language (source) and translates it to another language (target). There are mainly two different approaches to machine translation, 1) rule-based approach and 2) data-driven approach. In a rule-based approach the source language is analyzed thoroughly to identify some properties between the source and the target language. These properties when implemented as a set of rules help in translating the source language to the target language. The identified rules are responsible for transferring the grammatical structure between the two languages involved (source and target). Various tools such as morphological analyzers, parsers, taggers etc. are used to generate these rules. These rules can be identified one at a time by human experts or they can be identified by some kind of machine learning model. The second approach to machine translation is the data-driven approach, which makes use of large monolingual and/or parallel corpora[16] to translate the source language to the target language. The pre-requisite for this approach is a decent sized corpora as a source of knowledge. A statistical approach can be utilized to build a model out of the corpora to help in the translation.

A transliteration engine takes in letters of one language and maps it to the letters of another language. It attempts to produce a one-to-one correspondence between the languages involved (source and target). The mapping can be formulated based on any established ease of use methodology or based on matching sounds (phonetic approach) between the languages involved. The phonetic approach is widely used. Transliteration is used in cases where the target language script is not available, instead the source language script is used to represent the target language. *This work uses phonetic transliteration as an approach to translation for cross-language question answering task.*

1.1.5 General Evaluation Mechanism For QA Systems

From the previous sections it is evident that QA systems are not just made up of a single component but a series of components working together to achieve the final goal. Though the final goal of a QA system is to obtain a correct answer to the question asked, each of the individual components within the system has their own goals which eventually lead to the final goal. Thus the performance

16 A parallel corpus involves more than one corpora in different languages where each corpora is an exact translation of the others.

of the individual components are likely to influence the entire QA task. Ferrandez et al. [2006] says the overall accuracy of a QA system is directly affected by its ability to correctly analyze the question it receives. Moldovan et al. [2000] states that question analysis phase is responsible for 36.4% of the total number of errors in open-domain QA systems. QA systems may be evaluated in two different approaches. They are:

1. Black-box evaluation approach – Here the performance of the entire system is considered a whole without caring much about the performances of the individual components. Thus the final answer from the system is compared with the question asked to evaluate the QA system. A correct answer is what is expected from a QA system but there could be more than one correct answer for a given question. Thus the QA system needs to find all the possible final answers. Then there could be situations where the system needs to infer an answer from related facts. Further, for questions requiring a descriptive type answer it is hard to tell which answer is the best choice. Thus automatic evaluation methods and measures are not very suitable at all times to check the performance of a QA system. TREC is one such community involved in the research and evaluation of different tracks under the umbrella of information retrieval and extraction. TREC has some automated evaluation measures for QA systems but as TREC QA track is based on closed domains thus human judges play a major role in identifying the best answer for a question from the closed corpus and then allowing the researchers to compare their QA systems performance against those answers. Precision and Recall values are one way to evaluate QA systems performance and would fall under the black-box evaluation approach but the value itself might not make much sense in certain implementations.

2. Glass-box evaluation approach – Here each of the individual components of the QA systems are evaluated with appropriate methods particular to that component. The goal is to have optimal performances for each individual component which will ultimately lead to a better overall performance of the QA system. Examining and evaluating the components individually helps in identifying errors and problems particular to that component. Thus they can be fixed otherwise an error in one component leading to a poor performance in that particular component will be forwarded to the next component and so even if the next component was error-free due to some wrong input it might perform erratically and give bad results. This chain could follow resulting into an overall poor performance of the QA system. Thus glass-box evaluation approaches makes much sense to evaluate a QA system and fine tune individual areas.

Further details about these approach can be found with their implementation details at relevant sections of this document.

1.2 Cross-Language QA System

In a cross-language QA (CL-QA) system, the question are asked in language *A* and the system will look for the answers from a document collection which is in language *B*. After the system finds a relevant answer in language *B* from the document collection, the system will translate back that answer to language *A* (ideally) to finally present it to the user. In some implementations, the answer found in the intermediate language may not be translated back to the language in which the question was asked but rather left that way. The framework suggested in this work has a similar approach where the answer found in the intermediate language is not translated back to the language in which the question was asked and the reason behind such a stance is explained briefly in the next section as well as in further details in subsequent chapters. Thus the general architecture remains the same but with one or more additional translation components as shown in Fig 4.

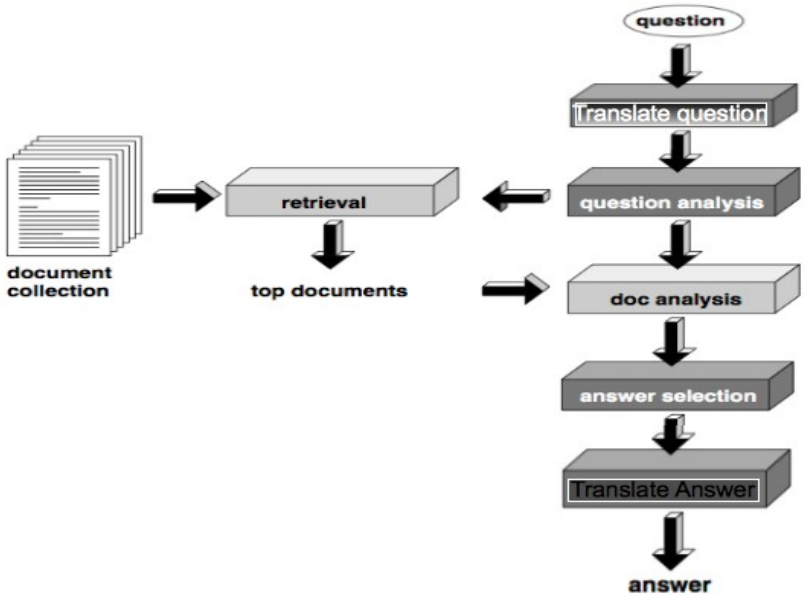

Fig 4: Architecture for Cross-language QA system [Monz 2003]

The translation component at the top takes in a question in language *A* and translates it to language *B*. The document collection is in language *B*. The actual QA system works only with one language that is language *B*. Once an answer (in language *B*) is found it is translated back to language *A* which is the actual output of the system. In the case of CL-QA system, the precision of the system depends on the correct translation and analysis of the questions that are received as input. An

imperfect translation of the question causes a negative impact on the overall accuracy of the system. Currently, there are four approaches in CL-QA systems to solve the bilingual task in which the question and the documents are in different languages [Ferrandez et al . 2006]. They are:

1. Using an automatic Machine Translation System to translate the question and the answer
2. Translating terms using a bilingual dictionary
3. Ranking results from different MT systems and choosing the best one
4. Using a set of pre-processed transformation rules in order to improve the translation outputs.

Examples of each of the approaches with their associated implementations are discussed in the relevant sections of the document.

1.3 The Bangla Language

The language Bangla or Bengali[17] is one of the Indo-Aryan[18] languages of South Asia with over 200 million native speakers. Bangladesh with a population of about 150 million is the largest concentration of Bangla native speakers. Bangla is also spoken in the western part of India.

Bangla is written in the Brahmi-derived Bangla script[19]. Bangla underwent a period of vigorous Sanskritization[20] that started in the 12th century and continued throughout the middle ages [UzZaman 2005]. The Bangla lexicon consists of tatsama (Sanskrit words that have changed pronunciation, but have retained the original spelling), tadbhava (Sanskrit words that have changed at least twice in the process of becoming Bangla), and a fairly large number of "loan-words" from Persian, Arabic, Portuguese, English and other languages. Also a large number of words are considered to be of unknown etymology.

The Bangla script is a segmental writing system where the vowel graphemes[21] are mostly attached

17 Interchangeably used with Bangla. From this point onwards only Bangla is used to refer to the Bangla Language.
18 The Indo-Aryans are the ethno-linguistic descendants of the Indic branch of the Indo-Iranians. As of today, there are over one billion native speakers of Indo-Aryan languages, most of them native to South Asia.
19 The Brahmi script, which appeared in the 5th century, represents the earliest post-Indus corpus of texts and some of the earliest historical inscriptions found in India. It is one of the most important writing systems in the world by virtue of its time depth and influence and is the ancestor to hundreds of scripts found in South, Southeast and East Asia. http://www.ancientscripts.com/brahmi.html
20 A particular form of social change found in India.
21 A fundamental unit in a writing language.

to the consonant graphemes as an ancillary glyph[22]. The Bangla script has a finite number of graphemes divided into vowels, consonants (including consonant clusters[23]), modifier graphemes, digits and punctuation marks. In the script 11 of the graphemes are vowels and 39 are individual consonants.

Fig 5: Bangla Graphemes (the ones in first row are vowels and the rest are individual consonants)

[http://www.itcfonts.com/Ulc/2611/BookRevBengaliChar.htm]

The Bangla script has an irregular phonetic nature, so apart from those 50 standalone graphemes, it can accommodate a large set of consonant clusters which ultimately create a gap between the phonetic and orthographic rules for a given Bangla word [UzZaman et al. 2006]. All these contribute to the complexity of the Bangla spelling rules with the Sanskritization process as the largest contributor [UzZaman 2005]. Despite all these complexities Bangla is the first language of choice for any sort of communication (written or spoken) among the native speakers. Numerous publications can be found in Bangla including text books, newspapers and official documents. The

22 A glyph is an element of writing. A grapheme is made up of one or more glyphs.
23 In Bangla consonant clusters are called *juktakkhors*.

use of Bangla was not very popular earlier in digital terms in the South Asian regions when personal computers were first introduced. The reason behind not using Bangla in daily computing was because of the complexity in spelling rules. Earlier there were no standardized keyboard layout for Bangla and thus proprietary fonts were only evolving with their own layouts. Each of such font developers mapped the Bangla letters to the English keyboard according to their wishes. This led to the problem of sharing a Bangla digital document more complex as both parties needed the same font installed to read a document. But, as time passed by, Bangla was included in Unicode[24] and further a standard keyboard layout was introduced to be followed for digital Bangla writings. With these introductions came Bangla Unicode fonts which made things much easier. Also localized versions of the popular softwares started to become available among computer users making Bangla the second language of choice in daily computing. Bangla fonts started to be available in handheld devices and mobile phones too. But the script having those 50 standalone graphemes and furthermore clusters and a complex set of spelling rules fails to attract more users to type using the Bangla script. Unless a user is very familiar with the standard Bangla keyboard, the Bangla typing process ends up to be very time consuming and error prone. But these situations did not hold native users from using Bangla for digital communications. Rather transliterated Bangla turned out to be much more popular in unofficial communications.

Transliteration is a way of mapping letters of one script to the letters of another script. Using a transliteration scheme, all those 50 standalone graphemes of the Bangla script can be mapped easily by the 26 letters of the Latin alphabet (English). Transliteration can be implemented by a letter to letter mapping (one to many correspondence possible too) between the English script and the Bangla script (both ways) and also based on the phonology of the letters of the target script. Users may key in their messages in Bangla using the English character set based on the original Bangla sounds. Such transliterated Bangla is exchanged over unofficial emails and text messages mostly. With the popularity of the use of Bangla in a transliterated form led to many digital applications in Bangla to evolve over this concept. Rather than asking users to type in Bangla, many application interfaces ask the user to key in their Bangla text in a transliterated form and the application maps the transliterated Bangla to an equivalent Bangla text using the Bangla script. And as English script is more accessible digitally, Bangla speakers tend to use such a transliteration scheme to express Bangla information more frequently.

24 Unicode is a computing industry standard for the consistent representation and manipulation of text expressed in most of the world's writing systems. The latest version of Unicode consists of a repertoire of more that 100,000 characters covering 90 scripts. http://en.wikipedia.org/wiki/Unicode

Despite the wide usage and rich diversity of the Bangla language, it lacks most of the language processing tools and resources specific to Bangla. The language is still in its infant stage as far as research in the area of computational linguistics is concerned. The Bangla language lacks a basic general purpose corpus as well as any computational grammars to parse Bangla sentences. There is some ongoing research to build some language specific tools for Bangla such as a news corpus [Arafat et al. 2006, Pavel et al. 2006], POS Tagger [Hasan et al. 2006], Text to Speech system [Alam et al. 2007], Bangla OCR [Hasnat et al. 2007], text summarization and categorization, machine translation system for Bangla, Bangla information retrieval systems and Head-driven phrase structure grammar for Bangla [Mahmud et al. 2007].

1.4 Aim Of This Thesis

This thesis discusses some of the cross-language question answering systems that are available mainstream. It highlights most of the important design and implementation issues of such cross-language question answering systems. Finally, with the gathered knowledge, a prototype framework is proposed for Bangla. Bangla has a huge speaker base but even then it lacks many of the basic computational resources and tools that are already available to other languages. The main research issue of this work was to explore the possibility of a QA system without having access to the mainstream components that are common to regular QA systems.

While designing the prototype framework for a language with very limited computational resources, many workarounds and limitations had to be accommodated because of the obvious reason of lack of proper resources specific to the language involved. However, based on the knowledge gathered from other systems a very limited capability interface for a Bangla QA system is built. The system is in no way a complete QA system, however, it gives a basis to implement a complete QA system for Bangla. The implementation involves questions from the medical domain only. The reason behind the choice of the domain is because the Bangla lexicon consists of a good number of loan-words from other languages. These loan-words sound almost the same as it would sound in its original language. And almost all the medical terms available in English have been imported to Bangla. Though some terms do have a proper translated version in Bangla, but the loan versions are used widely in daily basis. In this work a *translation based on transliteration and a table look-up method* is proposed as an interface to the actual QA task. The implementation part of this thesis thus involves transliterating a Bangla question as an equivalent Latin alphabet (English) version that could be used in an actual QA task.

The proposed framework discusses in detail how such an interface can help in a cross-language QA task. As stated already, the aim of this thesis is to give a detailed information on QA systems and mostly cross-language QA systems with an introduction to a new concept of *translation based on transliteration and a table look-up method as an interface to the actual QA task* where a proper machine translation engine is unavailable. Thus, *the work mainly explore the possibility of translation based on transliteration and table look-up as an interface for a limited domain QA task.* The performance of the proposed interface is evaluated and further details are given accordingly on how such an interface can help in developing a complete QA system for Bangla.

1.5 Chapter Summary

While this introductory chapter has presented some general details on Question Answering (QA) systems, the Bangla Language itself and the aim of this thesis, the following chapters will attempt to convey further details and approaches to the QA task. It will highlight the mainstream systems and concepts in details that are leading in the QA task and specially in cross-language QA task. This document further discusses the related works in Bangla language computing and how they have helped in bringing together the new idea of *translation based on transliteration and table look-up as an interface for cross-language QA system*.

The next chapter *Literature Review* highlights some of such recent important research related to cross-language QA systems and the hypothesis of this work.

The chapter *Design and Experimental Framework* discusses the proposal in further details with the design issues and arguments behind such approaches.

The chapter *Analyses, Evaluation and Discussion* discusses the overall system behavior after the prototype has been implemented.

Finally, the last chapter *Future Work and Conclusion* summarizes what we have learned and achieved through this work. It also gives directions to improvements, enhancements and future research.

2 Literature Review

There is much ongoing researches on QA systems and mainly on cross-language QA systems. This chapter is a restricted literature review of some systems (covered topic by topic) that somehow directly or indirectly influences in the main aim and the design decisions of the project.

From Pasca [2003] we have already learned that QA systems are basically made up of 3 main units which are the:

1. Question Processing Unit
2. Document Processing Unit
3. Answer Processing Unit

All these units may be produced individually by different groups for completely different purposes and later modified and merged to form the actual QA system. Thus each of the units may not have similar working patterns and internal design but still could achieve the final goal of question answering. The units could only be dependent on the outputs of their preceding units in the system flowchart, however, to achieve an overall optimal performance the units should work as closely and similarly as possible. There are certain overlaps between the units and thus if the units are designed as closely as possible then apart from the outputs of the unit other intermediate workings of the unit could help their neighboring units eliminating some redundant tasks.

2.1 Question Processing

By now we have already learned that though QA systems are somewhat similar to IR systems, they have different emphasis. Laurent et al. [2006] states 3 key features that identify a QA or an IR system. They are

1. The query mode is a natural language question for QA systems and keywords with some boolean operators for an IR system.
2. The output from the systems which can be an exact answer from the QA systems and a list of documents from the IR system.
3. The corpora which can be a closed and static set of documents or an open and dynamic document collection for any of the systems.

The input to a QA system is thus always a complete question. Bouma et al. [2006a] in their Joost system takes in a natural language question in Dutch. [Bouma et al. 2008], [Bouma et al. 2007], [Bouma et al. 2006a], [Bouma et al. 2006b] are actually a series of papers on the same Joost system that were published over several years highlighting the new things that have been added to the actual skeleton system. The system architecture of Joost is shown below.

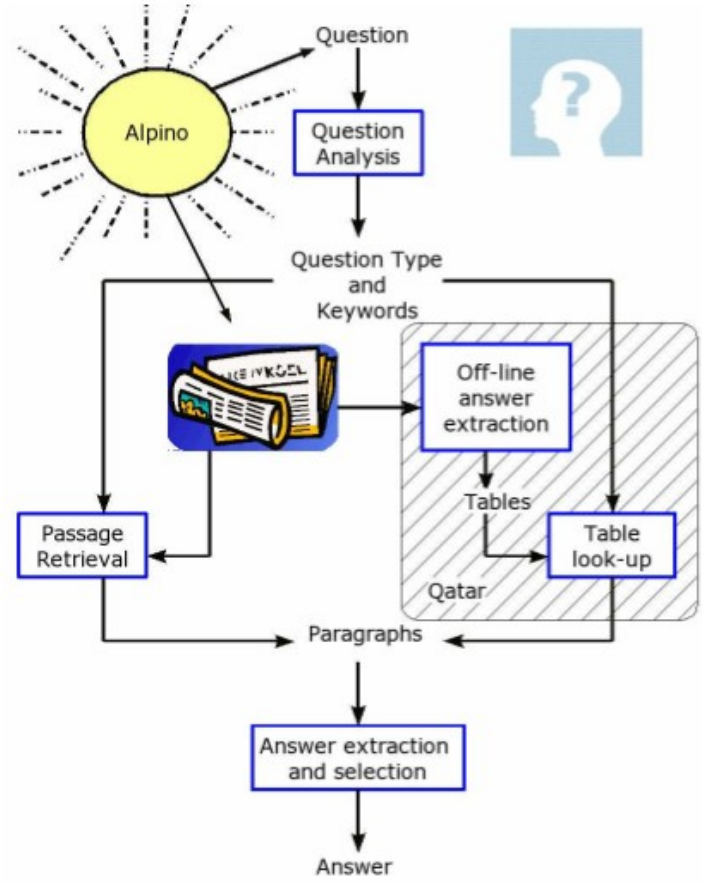

Fig 6: System architecture of Joost [Bouma et al. 2007]

Joost actually started as a monolingual QA system for Dutch. Once the question is given to the system the question is then parsed for syntactic information by a parser called Alpino[25] [Bouma et al. 2001]. Once the question is parsed by Alpino, the syntactic information is used to determine the subsequent steps of the entire QA task. The actual goal of this question processing unit is to

25 Alpino is a wide-coverage, linguistically-motivated grammar and parser for Dutch based on the HPSG formalism. It consists of 500+ grammar rules (defined using inheritance) and a large and detailed lexicon containing 100,000+ lexemes. Certain heuristics are implemented to deal with unknown words and ungrammatical or out-of-coverage sentences. The grammar provides a 'deep' level of syntactic analysis, in which WH-movement, raising and control, and the Dutch verb cluster (which may give rise to 'crossing dependencies') are given a principled treatment. The output of the system is a dependency graph [Bouma et al. 2005].

determine the question type and identify the keywords in the question [Bouma et al. 2006a]. Thus, on the basis of the dependency relations returned by the parser the question class is determined [Bouma et al. 2006b]. Joost works on a lot of pre-processed knowledge. This is a very common approach in QA systems where designers try to gather knowledge from the corpus beforehand to improve the actual QA task. Before actual questions are known, the text collection is exhaustively searched for potential answers to specific types of questions such as capital, abbreviation, dates etc. Such answers are extracted from the corpus off-line and stored in a structured table for quick reference during the actual QA task [Bouma et al. 2005]. Such off-line methods have proven to be very effective in QA [Fleischman et al. 2003]. The Joost system works on the CLEF[26] corpus. The entire CLEF data was thus parsed beforehand by the Alpino parser. Joost is able to determine 29 different question classes. 18 of those classes were determined by the off-line method. On the basis of the dependency relations returned by the parser the question class is determined in Joost. For each of the question class, one or more syntactic patterns were defined. Depending on the question classes, additional arguments can be identified.

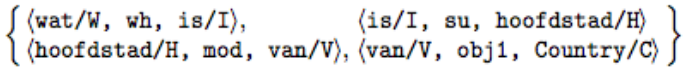

Fig 7: Identifying question class [Bouma et al. 2007]

The extract above (Fig 7) is a dependency parse from the Alpino. Here it is seen that the dependency relations assigned to the question "What is the capital of Togo?*"* (Wat is de hoofdstad van Togo?) match with the pattern in the figure and thus instantiate Country as "Togo". So the question class Capital is also assigned with "Togo" as an additional argument. Similarly, "Who is the king of Jordan?" would be classified as *function(king, Jordan)* and "In which year did the liberation war in Bangladesh take place?" would be classified as *date(liberation)*. Some question classes require access to lexical semantic knowledge which is obtained from the Dutch EuroWordNet like "In which American state is Iron Mountain?" asks for a location. Thus the system should be aware that 'state' refers to a location too. Further, "Who is the advisor of Yasser Arafat?" should be classified as *function(advisor, Yasser Arafat)*, so the system should know that advisor is a type of function [Bouma et al. 2006b]. Once the question type is determined the next stage of the Joost system is determined.

26 The Cross-Language Evaluation Forum (CLEF) promotes R&D in multilingual information access by (i) developing an infrastructure for the testing, tuning and evaluation of information retrieval systems operating on European languages in both mono-lingual and cross-language contexts, and (ii) creating test-suites of reusable data which can be employed by system developers for benchmarking purposes. **http://www.clef-campaign.org/**

As Joost works with a closed set of text such as the CLEF dataset, pre-processing is possible. But for systems working on open data set such pre-processing might not be possible at all.

Zhang [2004] is one such system where the questions are answered not from a saved collection of text but from the Internet in real-time. The system is called LAMP[27] which is based on the claim that the Internet is an ideal source of answers to a large variety of questions due to the fact that tremendous amount of information is available online these days. The system as others take in a natural language question, transforms it to an appropriate query and submits the formulated query to the popular search engine Google. The system is based on factoid questions only. It uses the Support Vector Machine (SVM) to classify the question given to the LAMP system. The system follows a two-layered question taxonomy having 6 coarse grained categories and also 50 fine grained categories [Li et al. 2002] like Table 4.

Coarse	Fine
ABBR	abbreviation, expansion
DESC	definition, description, manner, reason
ENTY	animal, body, color, creation, currency, disease/medical, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word
HUM	description, group, individual, title
LOC	city, country, mountain, other, state
NUM	code, count, date, distance, money, order, other, percent, period, speed, temperature, size, weight

Table 4: The coarse and fine grained question categories [Zhang et al. 2003]

The system assumes that questions are classified to a single category regardless of their ambiguity. They tried several other machine learning algorithms apart from SVM such as the Nearest Neighbors (NN), Naive Bayes (NB), Decision Tree (DT), and Sparse Network of Winnows (SnoW) to classify the questions to any one of those categories. They used two surface text features bag-of-words and bag-of-n-grams (all continuous word sequences in the question) in their SVM. The results show that SVM with the bag-of-n-grams feature had the most accuracy among all the other learning algorithms. Once it classifies the question it moves to formulate a query to be used. The process is discussed in the next section.

27 Learning and Answering Program (LAMP)

2.2 Document Processing

Once the question has been processed, the document processing unit becomes responsible to find relevant documents or passages related to the question given to the system at the beginning. The document processing unit is mostly a retrieval engine that takes in keywords and gives back passages or documents relevant to the keywords.

In Joost the question type determines the two possible ways of document processing. The Joost system has an off-line method called the Qatar component. This Qatar component answers those questions that match with one of the table categories. These tables are created off-line for facts that frequently occur in fixed patterns. These facts are stored, together with the IDs of the paragraphs in which they were found, as potential answers.

For those questions that cannot be answered by the Qatar component, a traditional keyword-based information retrieval is used to narrow down the search space for the linguistically informed part of the QA system which identifies the answer [Bouma et al. 2006b]. Agichtein et al. [2001] states that using search engine specific queries instead of the raw question might significantly improve the effect of question answering. Thus keywords are identified from the question using its content words. Irrelevant and function words are eliminated using a static stop-word list. The authors experimented with many publicly available IR engines and finally chose Zattair [Bouma et al. 2006b] as their IR engine because of the speed and recall performances. Using the keywords from the question, the IR system retrieves relevant passages from the corpus. The authors found through experimentation that segmentation of the documents into paragraphs is the most efficient for IR performance in a QA task [Bouma et al. 2006b]. They used existing markups in the corpus to determine the paragraph boundaries. Named entities found by Alpino were used as additional token to identify a paragraph and making the overall IR task easier.

In Zhang [2004] two ways are used to formulate the query.

- Interrogative Word Deletion: As question elements like “who”, “what”, “when” are usually not found in the answer thus they are dropped which increases the recall of the search without affecting precision. Regular expressions are used in the LAMP system to automatically remove such words.

- Verb Form Conversion: For instances like “When did Obama visit Afghanistan?” it is more likely that texts would be found as “... Obama visited Afghanistan ...” rather than as “... did Obama visit Afghanistan ...”. Thus LAMP converts the main verb from its original form to the third person singular form. The MEI[28] parser is used to locate the main verb in the question and PC-KIMMO[29] is used to find the different verb forms.

Once the query is formulated it is submitted to the search engine Google to obtain the answer. The top 100 search results are considered in the system to find the answer to the natural language question. In Google or any other Internet search engines, a search result is usually an URL[30], a title and some further string-segments of the related web document. Usually the title and the text segments are called “snippets”. The LAMP system uses only those snippets of text to find the answer which is described in the next section.

2.3 Answer Processing

At this stage any QA system would have some text snippets, passages or complete documents as candidates from where the actual answer is to be obtained. Systems may just present an exact text extract from a candidate as an answer or they may generate a proper answer from those candidate text segments. Some may give possibly a ranked list of answer candidates and leave it up to the user to make a pick or in ideal situations it would give a single well-formed correct answer to the question asked.

In the answer processing stage of Joost it has this far obtained a set of paragraph IDs either provided by Qatar or the IR system that was used. For questions that are answered by means of table look-up, the relation table provides an exact answer string. For other questions, answer strings are to be extracted from the set of paragraphs returned by the passage retrieval component. The paragraph IDs are used to retrieve the dependency relations of the sentences in those paragraphs. Bouma et al. [2006b] states that various syntactic patterns are defined to find the exact answer. For questions asking for a named entity the component should find a constituent headed by a word with the appropriate named entity. The authors claim that as all these occur frequently in the corpus, so more than one potential answer is identified from the text collection. Thus comes the need of ranking the potential answers. The authors also used the following features to determine a score for the answers:

28 A Maximum-Entropy-Inspired (MEI) Parser by Eugene Charniak

29 PC-KIMMO is a two-level processor for morphological analysis. It is designed to generate and/or recognize words using a two-level model of word structure in which a word is represented as a correspondence between its lexical level form and its surface level form. Available at http://www.sil.org/pckimmo/about_pc-kimmo.html

30 Uniform Resource Locator (URL)

- Syntactic similarity – the proportion of dependency relation matching with the question and the possible answers.
- Answer Context – a score is given for the syntactic context of the paragraph containing an answer.
- Names – the proportion of named entities found in the answer string.
- Frequency – the frequency of the answer in all the paragraphs returned by the IR engine.
- IR – the score assigned to the paragraph from which the answer was extracted.

Earlier the authors did not consider ranking the answers for table look-up method but later they implemented the entire ranking features to determine a score for all the possible answers.

Zhang [2004] is based on real-time data as whatever the search engine returns is new for the system. It uses a HMM-based named entity recognizer and some other heuristics to extract information from the snippets which would be the possible answer candidates. For the question "Who was the first American in space?" Google returns the following result.

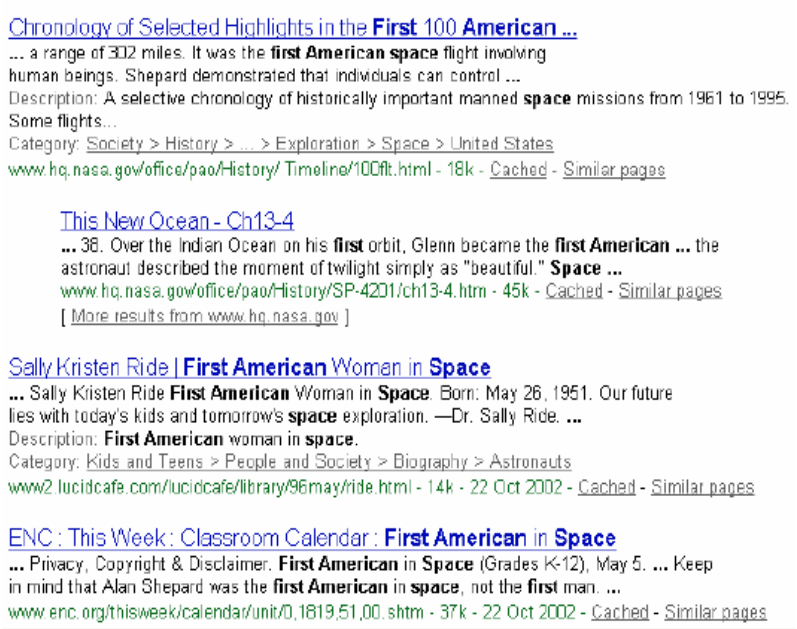

Fig 8: Google search results [Zhang 2004]

Thus the snippets extracted from Google after eliminating the other markups and URLs end up to be like the following.

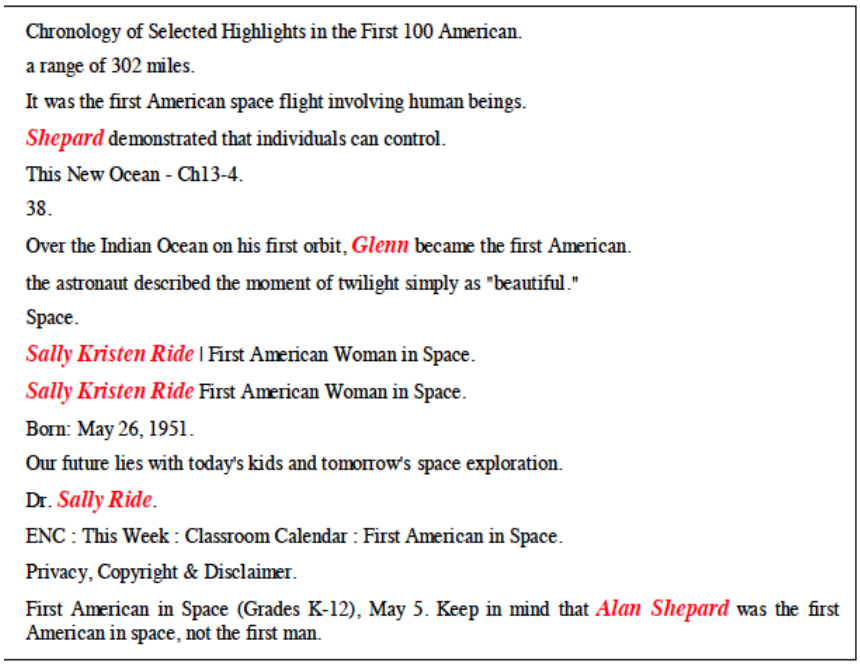

Fig 9: The snippets from Google [Zhang 2004]

Zhang [2004] describes a snippet as S containing a plausible answer A. Using bag-of-words feature vector $\mathbf{s}=(s_1,s_2, \ldots s_n)$, where n is the number of all words and s_i is the occurring frequency of the *i-th* word in snippet S. The question Q is also represented as a vector where $\boldsymbol{q}=(q_1,q_2, \ldots q_n)$. Each of the snippet S in the search result is assessed individually by the similarity between S and Q and the plausible answers contained in the top few snippets are selected. Zhang [2004] proposes an answer selection method based on aggregation. Thus for each plausible answer A the system aggregates all the snippets containing A into a cluster C_A. Also the snippet clusters of different answers referring to the same entity are merged into a single cluster as can been seen in the following example where *"*Sally Kristen Ride*"* and *"*Sally Ride*"* are merged to a single cluster as they are two variants of the same person name.

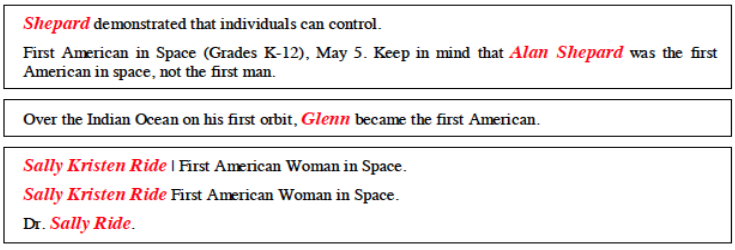

Fig 10: The snippet clusters constructed from the example [Zhang 2004]

The snippet cluster C_A of a plausible answer A summarizes A*'s* occurring context. It can also be

represented as a vector $\boldsymbol{a}=(a_1,a_2, \ldots a_n)$ where n is the number of all words and a_i is the occurring frequency of the *i-th* word in C_A which is equivalent to $\boldsymbol{a}=\Sigma_{A\varepsilon Sk}\boldsymbol{s}_k$. A score function is used to rank the plausible answers which is,

$$score(\mathbf{q},\mathbf{a}) = \|\mathbf{a}\|\cos\theta = \frac{\mathbf{q}\bullet\mathbf{a}}{\|\mathbf{q}\|} = \frac{\sum_i q_i a_i}{\sqrt{\sum_i (q_i)^2}}$$

where $\boldsymbol{q}$ is the feature vector of the question Q, $\boldsymbol{a}$ is the feature vector of C_A and Q is the angle between them. The function incorporates both similarity and redundancy information for answer selection and the value of *score(Q,A)* is the length of the "projection" of $\boldsymbol{a}$ on $\boldsymbol{q}$. All the plausible answers are ranked and the top ones are returned as an output from LAMP.

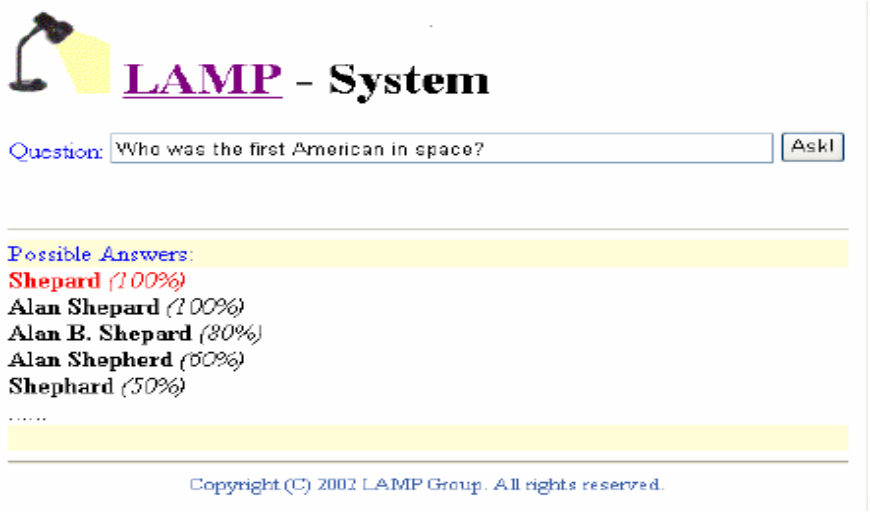

Fig 11: The LAMP system [Zhang 2004]

2.4 Evaluation Methods of QA systems

Laurent et al. [2006] talks about evaluation methods to compare performances and user-friendliness of both QA systems and Information Retrieval (IR) systems. They apply their methods of evaluation to Qristal (a French acronym meaning "Question Answering System using NLP") and Google Desktop Search engine. Qristal is based on the Cordial syntactic analyzer[31] and makes heavy use of all the usual constituents of natural language processing and sometimes manages to cover anaphora resolution and metaphor detection [Amaral et al. 2004]. The system evolved from a single-user program to a multi-lingual multi-user system [Laurent et al. 2006]. The latest marketed version of Qristal was evaluated in CLEF2005 and it ranked first in the evaluation for French as well as for all cross-language systems considering all the pairs [Laurent et al. 2005].

31 Cordial Analyser – Performs a morphological, syntactic and grammatical analysis of French texts. http://www.synapse-fr.com/Cordial_Analyseur/Presention_Cordial_Analyseur.htm

Laurent et al. [2006] evaluated the two systems on 3 different "user effort" criteria. They are:

1. the time needed to key in the question
2. the delay before the results are displayed
3. the reading time of the snippets or sentences to reach a correct answer.

The authors concluded that though IR systems have advantages in some "user effort" situations such as keying in the question, it is the the QA system that has an overall better performance over both the systems.

In the basic Joost system that the authors implemented, they claim that the scores were satisfactory for factoid and definition type questions. An extract of the scores are shown in Table 5.

Question Type	# questions	correct answers	
		#	%
Factoid	114	62	54.39
Temporally Restricted Factoid	26	7	26.92
Definition	60	30	50
Overall	200	99	49.5

Table 5: CLEF Evaluation for Joost

Bouma et al. [2005] found that out of the 140 factoid questions they had, 46 were assigned a type corresponding to a relation table. For 35 of those 46, an answer was located in one of the relation tables. The remaining 11 went through the IR component which was the fall-back strategy for the Qatar component. Parsing errors were the main cause of some wrong and incomplete answers. Bouma et al. [2006b] concluded from their implementation that the dependency parsing of both questions and the full document collection turns out to be very useful for developing an adequate QA system.

Zhang [2004] ran several experiments using the test questions and the answer patterns of the dataset from TREC-QA. They found that most of the TREC-QA questions can be answered from the snippets obtained from Google's top 100 search results.

dataset	q#	w#	percentage
TREC8	196	144	73.5%
TREC9	438	348	79.5%
TREC10	312	260	83.3%
TREC11	444	380	85.6%
total	1390	1132	81.4%

Table 6: How many answers to TREC questions can be found in Google snippets [Zhang 2004]

Zhang [2004] claim that the abundance and the variation of texts on the Internet allows the system to find correct answers with high probability, because the factual knowledge is usually replicated across the Internet in different expressing manners. The "Mean Reciprocal Rank (MRR)" and the "Confidence Weighted Score (CWS)" were used to rank their answers.

$$MRR = \sum_{i=1}^{n} (1/r_i)$$

For calculating MRR, n is the number of test questions and r_i is the rank of the first correct answer for the *i-th* test question.

$$CWS = \left(\sum_{i=1}^{n} p_i \right) / n$$

For calculating CWS, n is the number of the test questions and p_i is the precision of the answers at positions from *1* to *i* in the ordered list. The performance of the LAMP system on TREC11 dataset is shown in Table 7.

type	q#	MRR	CWS
PERSON	74	0.72	0.83
ORGANIZATION	15	0.56	0.49
LOCATION	101	0.56	0.65
DATE	95	0.65	0.81
QUANTITY	60	0.22	0.26
PROPERNOUN	53	0.39	0.59
OTHER	46	0	0
total	444	0.48	0.62

Table 7: MRR and CWS scores of LAMP [Zhang 2004]

The Mean Reciprocal Rank (MRR) score of LAMP was not close to the best QA systems in TREC and the author claims that this was due to the fact that the answer patterns (regular expressions) provided by TREC were very limited as many correct answers were judged wrong since they do not occur in the TREC specified document collection. Another factor they noticed was questions related to a period of time such as "Who is the U.S. president?" would be changing over time and can be

noticed over the Internet, however, in a closed document set the fact might not change at all over time. So, Zhang [2004] state that the text found over the Internet are messier than any closed document set such as the TREC document collection. Overall the LAMP system performed well on PERSON, LOCATION and DATE related questions. Zhang [2004] concludes by saying that a high performance QA based on Web search results is feasible.

2.5 Cross-Language/Multilingual QA systems

Multilingual support is a crucial aspect when the language of the search and the language of the text collection are different [Magnini et al. 2001]. Most QA systems work with a single language as it is much easier to implement. However, multilingual or cross-language QA systems can be useful in many scenarios. One such case would be where one of the language of interest does not have enough digital resources of its own to produce an answer to a question. In such scenarios the question could be given in language *A*, the question would be translated to language *B* as language *B* has more digital resources and finally the results obtained using language *B* would be translated back to language *A*. In some cases the answer obtained in language *B* may not be translated back to language *A* as will be the case of this prototype system proposed in this work. The actual scenario and the arguments behind the approach are discussed ahead. There are four different approaches to solve the bilingual task [Ferrandez et al. 2006] in any multilingual scenario. They are:

- Using a fully automated Machine Translation (MT) system to translate the question into the language in which the text collection is.
- Using a bilingual dictionary to translate word by word.
- Having a hybrid of automatic MT systems to translate questions/answers.
- Using a set of pre-processed transformation rules to translate questions or help in correcting translations of automated MT systems.

For the Joost system multilingual support was implemented too using an automatic MT system. The system took an English question, translated it into Dutch using the freely available translation engine Systran. There were some obvious drawbacks of using machine translation such as

- translations often resulted in grammatically incorrect sentences
- even if a translation could be analyzed syntactically, it contained words or phrases that were not anticipated by the question analysis module
- named entities and multiword terms were not identified or wrongly translated

Bouma et al. [2007] says that automatically generated translations are usually of poor grammatical quality. As their Joost system is based on parsing the question they found that due to poor grammatical form of the translated question there were unexpected parse results and thus the question classification is done incorrectly. To avoid such a situation they used an English question classifier based on the question classes of Li et al. [2002], the same classes used in LAMP by Zhang [2004]. For the monolingual part Joost only used the 40 question classes that they obtained from their Dutch dataset. They constructed a mapping from the question types used for English to the question types used in Joost. Both the mapped English question type and the Joost type assigned to the translations are used to find an answer to the question. Some mismatches were noticed in the mapping process as Bouma et al. [2007] claims that Joost expects a more fine-grained class than the class produced using Li et al. [2002]. For instance "What is the capital of Bangladesh?" would be classified as *loc:city* using Li et al. [2002] but as Joost has the class capital, it would thus be classified as capital. Bouma et al. [2007] further says that question classes assigned by Joost are not just labels but also includes some phrases from the question that eventually helps in answering the question. For instance a question like "What does UNICEF stand for?" would be assigned the label *abbreviation(UNICEF)* by *Joost* unlike other systems that classify the question as *abbr:exp*. In most cases the question classes assigned by Joost ended up to be more helpful than the classes assigned after mapping the English question class. However, exceptions were noticed for questions where Joost couldn't assign a class due to bad parsing because of grammatically incorrect translation. In such cases the use of mapped question class was more preferable over using no class at all. Later the authors used Wikipedia to improve the performance of their system. Wikipedia has a complex structure to hold the information. Here the authors removed some irrelevant materials from the original XML version of the Dutch Wikipedia and finally used a highly simplified XML version that contained only the information that were enough to identify the segmentation of the text into titles, sections and lists [Bouma et al. 2007]. Further as Wikipedia keeps on expanding and new things come into being, the authors started using the templates to identify the basic information for a given entity using the list of attribute-value pairs [Bouma et al. 2006a]. They used XQuery to extract all the attribute-value pairs from all the templates present in the Dutch Wikipedia. About 1.3 million tuples of the form <*object, attribute, value, template_name*> were found i.e. <*AFC Ajax, stadion, Amsterdam, ArenA, Voetbal_club_infobox*>. The authors claim that the information in the template tuples were potentially very useful for the QA task.

Further the authors considered expanding the query for a better IR performance. They tried to

extract and add various lexico-semantic information to the query such as:

- nearest neighbors from proximity-based distributional similarity
- nearest neighbors from syntax-based distributional similarity
- nearest neighbors from alignment-based distributional similarity

Bouma et al. [2007] concludes by claiming that the inclusion of Wikipedia made the QA task more realistic and attractive. They are considering on using the structure of Wikipedia more seriously which would enable answer extraction that combines NLP with XML-based extraction.

2.5.1 Components That Are Used To Give Multilingual support

To implement full or partial cross-language or multilingual support to any systems including QA many approaches could be undertaken. An automatic machine translation engine between the languages involved would be an ideal choice for that. A simple dictionary lookup method can also be employed to translate texts between languages. Other heuristics could be used too such as Jiang et al. [2007] suggest a transliteration approach with web mining to improve the named entity translation. Such an approach could be very useful in QA systems as most questions include some form of named entities within them. Jiang et al. [2007] suggest a 3-level transliteration model, 1) English surface string to Chinese Pinyin[32] string, 2) Chinese Pinyin string to Chinese character string and 3) Chinese character language model. For a given English named entity, denoted as *E*, Jiang et al. [2007] syllabify it into a syllable sequence $SE = \{e_1, e_2 \ldots e_n\}$ with some linguistic rules stated in the paper. For example, *"*Clinton*"* is split into *"*C / lin / ton*"*. Then a generative model is used to transliterate the syllabified English name into Chinese character string based on Knight et al. [1998]'s Machine Transliteration System. For the generated "syllable" sequence $SE = \{e_1, e_2 \ldots e_n\}$ a Chinese character sequence $C = \{c_1, c_2 \ldots c_m\}$ is looked for with the criteria $C^* = argmax\ p(SE|PC)\ p(PC|C)\ p(C)$ where *PC* is a Chinese Pinyin sequence, $p(SE|PC)$ is the probability of translating *PC* into *SE*, $p(PC|C)$ is the probability of translating *C* into *PC* and $p(C)$ is the generative probability of a character-based Chinese language model. The transliteration model is evaluated by the Edit Distance measure between the character sequence of the "correct" transliteration and the character sequence output by the system. [Jiang et al. 2007] claims the addition of a transliteration model in NE translation improved the precision and recall of the NE translation by a large margin. In their sample of 50 NEs, 48% were correctly translated.

32 Pinyin is the most commonly used romanization system for Standard Mandarin. http://en.wikipedia.org/wiki/Pinyin

Finch et al. [2008] presents a technique for transliteration based directly on techniques developed for phrase-based statistical machine translation. They obtained correct or phonetically correct results 80% of the time where the focus was to use transliteration to translate unknown words in a speech-to-speech machine translation system.

UzZaman et al. [2006] propose a comprehensive English to Bangla transliteration scheme to handle the full complexity of the Bangla Script. A phonetic encoding scheme is proposed to produce an intermediate code-string that facilitates matching pronunciations of input strings and the desired outputs. The proposed system has two approaches, a direct phonetic mapping and a lexicon enabled mapping.

All these transliteration approaches can be used in cross-language QA system scenarios. This thesis work proposes such a *transliteration based approach to translation along with a table look-up method as an interface in a cross-language QA system scenario*. The language under consideration in this thesis work is *Bangla*. Thus some *Bangla* machine translation related literature are discussed too.

In Dasgupta et al. [2004], a 5-stage transfer based architecture is proposed to obtain a Bangla syntactic tree from an English syntactic tree with an optimal time complexity for an English to Bangla machine translation system.

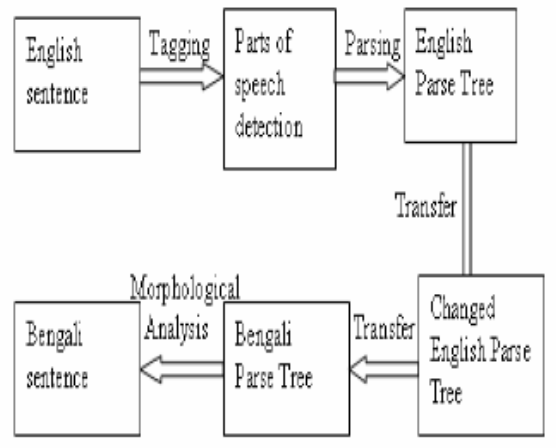

Fig 12: Proposed transfer architecture for MT (Dasgupta et al. 2004)

The 5 stages are 1) Tagging, 2) Parsing, 3) Changing the CNF[33] parses to normal ones, 4) Transferring English trees to equivalent Bangla trees and 5) Generating morphological analysis. The authors used the CYK[34] algorithm and claimed that the parsing steps were minimized to polynomial order from exponential order. The CNF parses were converted to a normal parse tree using some transformation rules and finally the transformed parses were converted to Bangla parse trees using a bilingual dictionary.

Hossain [2008] developed an open-source English to Bangla machine translation system called Anubadok[35]. It is written in Perl and uses the Penn Treebank annotation system for natural language processing. It uses four major steps in translating from English to Bangla: 1) Pre-processing of English documents, 2) POS tagging of documents from step 1, 3) English to Bangla translation of POS-tagged documents and 4) Post-processing of translated documents. The POS tagged English words are translated using a bilingual dictionary and then the translated words are organized in the usual Bangla syntactic order (SOV[36]) to produce the final translation. The system writes out translated documents as Unicode encoded Bangla texts.

2.5.2 Finite State Methods

Apart from automatic machine translation engines many other approaches could be used to obtain a translation. These approaches could be used alongside the automatic machine translation engines to improve their performance or they could be used completely on their own to translate text between languages. Though an automatic machine translation engine is more preferable to translate texts between languages, however, for some language pairs an automatic machine translation engine might not be available. Bangla is such a language which has limitations in its digital language processing tools. A complete Bangla to English machine translation system is yet to be available though there are some ongoing initiatives. This thesis work deals with the language Bangla and proposes *an approach to translation based on transliteration* in a cross-language QA system scenario. The translation based on transliteration is achieved using the popular finite state technology. Thus some important and related literature regarding finite state methods and technologies are highlighted here.

33 Chomsky Normal Form
34 Cocke-Younger-Kasami
35 Anubadok is a Bangla word which literally means the one who translates.
36 Subject-Object-Verb. Though Bangla has a relatively free word-order SOV is the most common form.

An automaton, a mechanistic device, can be designed to embed certain properties of a formal language. A formal language is a set of strings made by concatenating together symbols taken from a finite vocabulary. The language may comprise a finite or an infinite number of sentences. For a finite number a complete list of the sentences can be written down. But if the language generates infinite number of sentences then it is not possible to list all the possible sentences, however, a grammar can be defined which can characterize the sentences in some form of recursive or iterative manner. Such grammatical rules can be applied to either produce further sentences or to recognize certain sentences. Regular expressions are the most handy way to express such regular languages. Regular expression is basically a formula that embeds the rules in which the symbols can be used within a string. Regular expressions can be easily converted to a particular kind of automaton called the finite-state machine which can be used to generate or check for the consistency of an input string based on the actual grammatical rules formulated earlier. A finite-state machine consists of a finite number of states and a function that determines transitions from one state to the others. The machine somewhat represents the process of reading a sequential tape. The machine starts at a distinguished initial state with the tape positioned at the first symbol of a particular string. The machine transitions from state to state as it keeps reading the tape and eventually exhausting the input tape. At the end of the tape if the machine is found to be in one of the states designated as the final state then the machine has accepted the string read from the tape otherwise not. A finite state machine is represented as a state-transition diagram where circles are the different states and arcs between the circles are the transitions. The start state consists of an arrow pointed towards the state and the final states are enclosed with double circles (Fig 13).

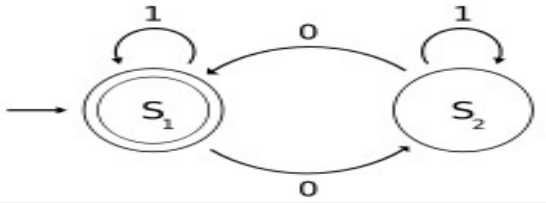

Fig 13: A finite state machine
[Wikipedia]

Transducers are a special type of automata which ultimately generates an output string. Each of the transitions in the automata are labeled with two symbols. One of the symbol represents input and the other represents the corresponding output. The transducer translates the input string to an output string (Fig 14).

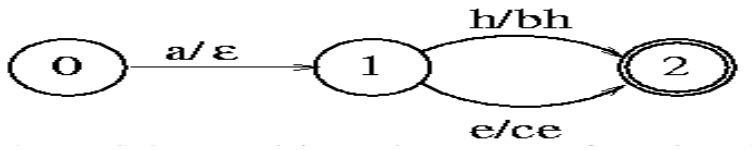

Fig 14: A finite state transducer [Jan Daciuk]

Though the syntax of a natural language cannot be completely described using finite-state machines and regular expressions, however, the mathematical and computational simplicity of regular expressions and finite-state machines can be used in different contexts to describe certain properties of a natural language in a simple manner. Regular expressions have a clean, declarative semantics but at the same time they constitute a high-level programming language for manipulating strings, languages and relations [Kartunnen 2000]. For this reason they have turned out to be very useful for linguistic applications. Descriptions consisting of regular expressions can be efficiently compiled into finite-state machines which can eventually be determinized, minimized, sequentialized, compressed and optimized to reduce use of computational resources as well as time.

2.5.3 Popular Finite State Manipulation Tools

This project makes use of finite-state machines specially transducers to the fullest to achieve its primary goal of *translation based on transliteration*. The actual design of the finite-state transducer is described in the next chapter. Here all the popular tools to handle finite-state machines are described.

The Finite State Compiler (FSC) is an interactive interface for finite state calculus developed by [Tapanainen 1995]. Apart from converting simpler regular expressions into finite state automata it is also able to handle extended regular expressions, allows sophisticated features like lexical lookup and analysis, parsing, writing scripts etc. It can further handle alternative regular expressions for writing idioms. FSC was built primarily to handle multiword expressions (MWE) which could take different forms while it is being identified from within a text. Those multiword expressions are encoded as regular expressions according to a developed notation. Each idiom is compiled into a finite-state network [Segond et al. 1995]. Whenever an idiom is matched their corresponding meanings can be obtained from the transducer.

Xerox Finite-State Tool (XFST) [Beesley et al. 2003] is a utility tool to handle finite-state networks.

It is developed by the Xerox Corporation[37]. Simple automata and transducers can be easily created using XFST. It is a successor of two similar earlier implementations IFSM and FSC. XFST is able to read finite-state networks from binary files, regular expressions and other networks by a variety of operations. It uses virtual networks to avoid excess computation which is a primary drawback in traditional finite-state operations which produce huge networks. Users apply a network to determine whether the string is accepted by the network or transform it to another string if the network is built as a transducer. The tool allows different ways to get information about the virtual network and finally to inspect and make modifications in that virtual structure.

FSA Utilities toolbox [Van Noord 1997] is a collection of utilities to manipulate regular expressions, finite-state automata and finite-state transducers. Using the toolbox it is possible to construct automata from regular expressions, performing minimization, composition, complementation, intersection, Kleene closure, determinization (both for finite-state acceptors and finite-state transducers) etc. The toolbox is available under the GNU General Public license and allows various visualization tools to browse finite-state automata. FSA supports four types of automata:

- recognizers
- weighted recognizers
- transducers
- weighted transducers

It can also handle macros and other user-defined regular expression operators. One further advantage with FSA toolbox is that it can produce C and Prolog code of a finite-state automata or finite-state transducer to be used in other implementations.

Except for FSA toolbox all the other tools named above are proprietary and not freely available.

2.6 Summary And Proposal

All the researches highlighted this far implement many different technologies related to QA systems and specifically Cross-Language QA (CLQA) systems. None of the systems mentioned this far can be considered a complete solution to Question Answering; however, each one of them has addressed

37 http://www.xerox.com/

some particular issues in QA that make them perform better in particular situations. Compared to IR, QA systems are still in the early stages of research and thus only a handful of QA systems have emerged this far. The already available QA systems are maturing day by day and implementing more features to address a wider variety of issues in question answering. Most of the QA systems available these days are usually for languages with a large amount of digital resources and having a good number of matured language processing tools. Thus with the availability of such language processing tools and resources, the process of building a QA system becomes much easier as a QA system is made up of several components utilizing such varied tools and resources. But for languages with limited digital resources and processing tools the entire process of designing and implementing an all purpose Question Answering system turns out to be very difficult. Some assumptions and considerations have to be made in an attempt to design just a basic QA system for languages with limited resources.

Bangla, though being being one of the top 10 most widely spoken languages with over 200 million speaker, is one such language with very limited digital resources and language processing tools. The language is still in its infant stage as far as research in the area of computational linguistics is concerned. The language lacks the very essential general purpose corpus to be used for different Bangla language processing tasks. There are some ongoing initiatives to build a large general purpose corpus and already a 97 million word electronic corpus of South Asian Languages is available from the EMILLE[38] project which includes Bangla. Further there is also a News corpus of Bangla developed from the articles of the online version of a popular Bangla newspaper of Bangladesh. Also there are ongoing researches on many different language processing tools for Bangla; however, no notable research on Bangla Question Answering systems can be found till date.

Building a complete open-domain QA system for Bangla is not yet feasible as there are not many digital texts available in Bangla on varied topics. Large collection of Bangla texts are available in non-digital format but that doesn't help much in a digital QA system. At this point a cross-language QA system can be very effective for the Bangla language. There are significant amount of digital texts available in other languages, specially in English on varied topics which can be used to answer Bangla questions. A system can be designed to take a Bangla question, look for the answer in an English text collection and later the answer can be translated back to Bangla to present to the user.

38 Enabling Minority Language Engineering (EMILLE). Available at http://www.emille.lancs.ac.uk/

The proposal sounds reasonable but as the language Bangla lacks many language processing tools, the proposal comes with *a bottleneck of translating the text between Bangla and any other languages*. Bangla is yet to have a complete machine translation system. From earlier texts we have learned that there has been some work on Bangla MT systems but none of them are complete on its own to be used as a component to aid in cross-language QA systems.

This thesis has studied and reviewed many of the challenges to be met in building mono-lingual as well as multi-lingual QA systems. With the knowledge gathered from reviewing those available systems an attempt is taken to design a small scale QA system for Bangla which depends upon English text and is limited to a particular domain. As already mentioned, Bangla lacks many forms of digital resources, and thus a complete general purpose QA system for Bangla would not be possible unless many other tools and resources become available. And it is definitely beyond the scope of this dissertation to present an all-purpose Bangla QA system. However, a prototype Bangla QA system is designed and partially implemented, based on the Joost and LAMP systems discussed earlier. The proposed system uses a type of transformation rules, one of the four types of approach to translation, to partially translate Bangla questions to its equivalent English and then searches over the Internet to look for potential answers. The partial translations are achieved using transliteration based on finite state technology. The concept of transliteration in QA systems is relatively under-explored, let alone translation based on transliteration. Along with the transliteration module a table look-up approach is also employed to obtain an English question from a transliterated Bangla version. The next chapter discusses in detail the design of such a cross-language QA system which takes ideas from already existing QA systems and uses some Bangla language specific phenomenon to solve a very limited scale Bangla QA task.

3 Design Of The Experimental Framework

The previous chapters have given a detailed overview of the technologies related to Question Answering (QA) systems as well as Cross-Language question answering (CLQA) systems. This chapter introduces a framework for such a cross-language QA system where one of the languages involved has limited digital resources. The proposed design uses concepts from existing state-of-the-art systems but, due to limited language resources and overall time allocated for the project, a limited scenario of the question answering task is addressed.

3.1 Background

Transliteration is a way of mapping letters of one script to letters of another script. Using a transliteration scheme all the 50 standalone graphemes of the Bangla script can be mapped easily by the 26 letters of the Latin alphabet (English). It can be implemented by a direct *letter to letter* mapping (one to many correspondence too) between the English script and the Bangla script (both ways) and also *based on the phonology of the letters* of the target script. The second form of transliteration (transliteration based on the phonology of the letters) is easier to formulate and thus is more popular. Users key in their messages in Bangla using the English character set based on the original Bangla sounds. Such transliterated Bangla is exchanged over unofficial emails and text messages mostly. The popularity of the use of Bangla in a transliterated form led to many digital applications in Bangla to evolve over this concept. Many application interfaces ask the user to type in their Bangla text in a transliterated form and the application maps the transliterated Bangla to an equivalent Bangla text using the Bangla script. And as English script is more accessible digitally, Bangla speakers use such a transliteration scheme widely to express Bangla information more frequently.

e.g.

Message in English	*My name is Nafid Haque.*
Message in Bangla Script	আমার নাম নাফিদ হক
Transliterated Bangla	amar nam nafid hq a|ma|ra na|ma na|fi|da ha|ka
Gloss	*my name nafid haque*

Table 8: Bangla Transliteration Example

The concept of transliteration has been exploited in many ways to make use of Bangla digitally. There are interfaces available for the web and mobile devices that take in Bangla text in a transliterated form and produce the same text in the original Bangla script. The concept of the use of Bangla in a transliterated form is exploited further in this thesis work to translate some Bangla words to equivalent English versions.

3.2 Proposal

As stated earlier, the Bangla lexicon consists of a good number of "loan-words" from Arabic, Persian, English and other languages. And most of them are pronounced almost the same way as would be pronounced in the original language.

Following are some *English* words that are pronounced almost the same way in Bangla.

Police, Telephone, Television, Computer, Table, Chair, Bottle, Bus, Truck, Train, Ulcer, Cancer

The following table gives a detailed comparison:

English Word	Actual Bangla Spelling	Transliterated Bangla
Police	পুলিশ	pulish
Telephone	টেলিফোন	Telifon
Television	টেলিভিশন	Telivishn
Computer	কমপিউটার	KmpiuTar
Table	টেবিল	Tebil
Chair	চেয়ার	cheyar
Bottle	বোতল	botl
Bus	বাস	bas
Truck	ট্রাক	T\rak[1]
Train	ট্রেন	T\ren
Ulcer	আলসার	alsar
Cancer	ক্যানসার	k\zansar[2]

Table 9: Bangla Transliteration Example

From Table 9 we notice that the transliterated Bangla is very similar to its equivalent English

[1] '\' is used to produce a consonant cluster between 't' and 'r'.

[2] '\z' is used to produce a phonetic emphasis on the previous consonant.

versions but not exactly the same. As Bangla has more characters and supports more phonemes[39] than English, to accommodate the actual pronunciation of such loan-words in Bangla, the transliterated Bangla version of those loan-words are not exactly spelled the same way as they are in English. However, the transliterated Bangla version, which we may call as pseudo-English version, can be intelligently processed to get back the original English spellings. *This thesis explores that possibility.*

As stated earlier most of these loan-words in the Bangla lexicon sound almost similar to the ones in their original languages but that does not necessarily mean that those loan-words do not have an exact Bangla translation. The words "television" and "telephone" both have Bangla versions like "duro-dorshon" and "dur-alaponi" respectively but "Telivishn" and "Telifon" are more commonly used in every day official and unofficial communications. There is no single specific genre to which these loan-words can be categorized like only electronic names or auto-motives but it is noticed that, in the Bangla lexicon, almost all the medical terms are such loan-words and sound exactly or similar to the imported form. Thus, in this thesis we limit our experiment to only those medical terms and test the hypothesis of *translation based on transliteration* as an interface for a limited domain cross-language question answering system scenario. The entire work can be divided into two components, the translation based on transliteration with table look-up and the question answering part.

Fig 15: Components of the proposed system

In this thesis the first component is exploited in detail and a small prototype system is developed. The second component has been explored mostly theoretically and thus the findings of the research will be stated.

3.3 Design

The basic idea of the system is to 1) take in a question in Bangla written in a transliterated form, 2) translate that Bangla question to its equivalent English version, then 3) search over the Internet for

39 A phoneme is the smallest contrastive unit in the sound system of a language.

the answer to the question, 4) translate the English answer to Bangla and present it to the user. As the goal of this project is not to develop a complete Machine Translation system for Bangla, the answer/result obtained in English for the given question will not be translated to Bangla. *The work concentrates mostly on getting the English version of the question from its Bangla equivalent using translation based on transliteration and table look-up.*

These days a wide variety of information is available on the Internet and a good amount of it are in English. With the introduction of web blogs and forums, any individual can post a view or an article, thus contributing to this huge information pool. Initiatives are taken to make all the published books available over the Internet. These days electronic articles, magazines and books are more popular than their printed versions. The electronic versions are easily accessible, cheaper than printed ones, takes less or no physical space at all and most importantly can be digitally searched and processed. Thus, while looking for specific information, rather than buying a book or a magazine from a shop, the trend has become to search for the topic over the Internet and obtain all the related information at one place without even going through unnecessary content. An individual looking for legal information can avoid going through a huge pile of books on law but just search over the Internet to obtain some information about his or her query. The same is the case for an individual wanting to know more about a disease or a medical term. The person can have some basic idea about the disease from all the medical texts available over the Internet without consulting the medical books. *These uses of the technology are not meant to eliminate a medical doctor or a legal advisor from the society but the access of information through these technologies is meant to make individuals better informed rather than keep them totally ignorant of the basic information.* People having access to basic information on a variety of topics over the Internet can make a small research of their query and have a basic background before they move to seek professional advice. Information retrieval systems in the form of Internet Search Engines have already made it possible for people to have access to basic information. These search engines take in query terms and point to documents having information about those terms. However, this trend has moved towards complete question answering, where a user asks a complete question and expects a complete and correct answer in return.

The design of this framework is highly motivated by the LAMP and the Joost systems discussed in the earlier chapters. The LAMP system by Zhang [2004] claims that the Internet is an ideal source of answers to a large variety of questions due to the fact that a tremendous amount of information is available online these days. The information available online is written in many different languages

and a huge share of this information is in English. There are texts available in Bangla too over the Internet but the volume is still not comparable to many of the other most spoken languages of the world. Also, the Bangla texts available online are mostly limited to the news genre as online Bangla newspapers are very popular. So these limited Bangla texts are not enough for Zhang [2004]'s claim that online resources are an ideal source of answers to a large variety of questions for a Bangla QA system scenario. But if the English content available online is considered as the search space to answer Bangla questions, then a large variety of questions could be answered. Having such a cross-language solution could benefit many native speakers of Bangla as they could ask a question in Bangla and obtain their desired information. The proposition sounds feasible but comes with the *bottleneck of translating a Bangla question to the equivalent English*. This is achievable if a complete automatic machine translation system is available between Bangla and English in both directions. Then a Bangla question can be easily fed to a translation engine to obtain an English equivalent question, and that obtained English question can be processed like the Zhang [2004]'s LAMP system or Bouma et al. [2006]'s Joost system to obtain the answer. Lastly that answer can be translated back to Bangla for the user. But as stated earlier Bangla is still to have a complete Bangla/English machine translation system, making the proposal of a QA system for Bangla to be hard. Thus, to create such a cross-language QA system for Bangla, the first step is to build a complete Bangla/English MT system, which is another research area of its own. But until such a complete translation system becomes available, the cross-language QA task for Bangla maybe be solved using a slightly different approach, limiting some features and options of a full-fledged QA system.

From earlier discussions we have noticed that Bangla words have some complex phonetic and spelling structures. These complexities of the language can be utilized to translate specific Bangla words to equivalent English forms. If this property of the Bangla language can be generalized and automated, then a very limited scale translation system can be produced which may ultimately help in a limited scenario QA task. We have mentioned already that the Bangla lexicon consists of a good number of loan-words from other languages that are pronounced somewhat similarly to the sound of their original languages. Though these loan-words are spread around among different genres, the medical terms take a big share in the list of these loan-words. Table 10 highlights some such medical terms.

Medical terms in English	Transliterated Bangla
cancer	*kansar*
heart attack	*hart etak*
fracture	*phrakchar*
stroke	*estrok*
liver	*leevar*
lung cancer	*laang kansar*
blood	*blad*
conjunctivitis	*konjunktivitis*
fever	*fivar*
cyst	*sist*
flu	*flu*

Table 10: Medical Terms

From Table 10, if we try to speak out the transliterated Bangla versions of these medical terms, we will be more or less able to guess the actual English versions. This is due to the fact that these terms have been imported into the Bangla lexicon but due to many language specific properties especially phonetics, the pronunciation has changed slightly. The original versions are broken into smaller syllables and then converted to the phonemes available in Bangla to pronounce. Table 11 gives an overview of that property.

English	Bangla
can + cer	*kan +sar*
heart	*ha +r + t*
a + tack	*e + tak*
frac + ture	*phrak + char*
stroke	*es + t +ro + k*
li + ver	*lee + var*

Table 11: Terms broken to syllables

From the above comparisons, a technique can be devised to transform the syllables of the transliterated Bangla to the syllables of the original English version. If that transformation can be performed as closely and correctly as possible, then we are able to obtain the original English spellings from the transliterated Bangla forms. The approach is similar to Jiang et al. [2007], who

suggest a transliteration approach to improve the named entity translations. Jiang et al. [2007] syllabify an English named entity into a syllable sequence like the word *"*Obama*"* to *"*O / ba / ma*"* and then use a generative model to transliterate the syllabified English name into a Chinese character string. They used this approach in their QA task with the claim that most questions include some form of named entities within them and implementing such a transliteration model improved their named entity translation. For a Bangla QA scenario for the medical domain, the medical terms are the named entities which can be translated to English using a transliteration model like Jiang et al. [2007]. In our case a syllabified transliterated Bangla (pseudo-English) term is transformed into a correct English term to be used for further processing.

This *transliteration approach to translation* is very similar to the morphological analysis of a word. A word is typically a stem together with a set of affixes. The smallest meaning-bearing units are called morphemes. During the analysis, morphemes are identified. Finite-State technologies are widely used for morphological parsing.

Input	Morphological Parsed Output
cat	*cat + N + SG*
cats	*cat + N + PL*
books	*book + N + PL*

Table 12: Morphological Analysis [Jurafsky et al. 1999]

The morphological parser has knowledge about the

- lexicon – the list of stems and affixes
- morphotactics – the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word
- orthographic rules – the spelling rules of the words

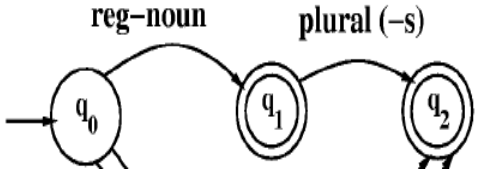

Fig 16: A simple FSA for English nouns [Jurafsky et al. 1999]

The finite-state automaton above accepts regular, orthographic singular and plural English nouns. Further, a two-level morphology model as finite-state transducers.

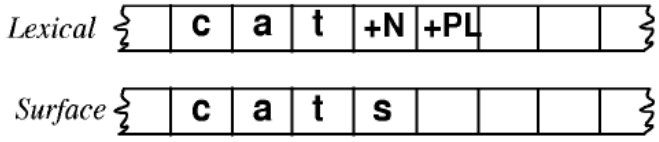

Fig 17: Two-level morphology [Jurafsky et al. 1999]

The finite-state transducer is able to map between the lexical and the surface level of the words.

A similar two-level approach could be used in the *translation based on transliteration approach* for the translation of the medical terms of Bangla. Individual syllables are dealt with instead of the morphemes.

Terms		**Individual Syllables**
Transliterated Bangla	*kansar*	*kan + sar*
English	*cancer*	*can + cer*

Table 13: Two-level approach to transliteration

A finite-state transducer maps the syllables of the transliterated Bangla form to the English ones. Once all the English syllables have been obtained correctly and merged together it is possible to get the correct English word. This approach should work for most of the medical terms if designed very carefully. But a QA scenario involves not just a single term but a complete question. Now the question is whether such a two-level transliteration approach works for an entire question given in a transliterated Bangla form. Unfortunately the answer is negative. As we have mentioned earlier, the Bangla lexicon has a good number of loan-words that sound similar to their original versions but the rest of the words in the lexicon are not such loan-words. And these Bangla words have no phonetic similarity with their English counterparts.

English	**Transliterated Bangla**
book	*boi*
car	*gari*
where	*kothay*
why	*keno*
here	*ekhane*
how	*kibhabe*

Table 14: Literal translations

Table 14 includes some question elements too which would occur in a typical question scenario.

e.g

English:	*What is Cancer?*
Translation in transliterated Bangla:	*Kansar ki?*

The two-level transliteration approach that we were suggesting earlier only works for the named entity in the question but does not work for the other terms in the question. And if we only work with the named entities, then the actual purpose of question answering is not achieved; rather the system would end up as a cross-lingual information retrieval (IR) system.

e.g

English:	*How do you treat Cancer?*
Translation in transliterated Bangla:	*Kansar kivabe chikitsha korte hoi?*

In the previous two example questions, if the medical term *"*Cancer*"* was only considered then for both the cases the IR engine will produce identical results. But "What is Cancer?" and "How do you treat Cancer?" are two completely different questions with different set of answers. What differentiates between the two questions is not the named entity or the medical term but the other terms in the question. Thus, to attempt a QA task the entire question needs to be considered. So, now the issue is to map the transliterated Bangla word "ki" to English "What is" or "Kivabe chikitsha korte hoi" to "How do we treat". This would be easily possible if a bilingual dictionary was available. Hossain [2008]'s Anubadok system is an English to Bangla MT system which uses a bilingual dictionary to translate POS tagged English words to Bangla. But for our QA task, we need a Bangla to English MT system or at least a Bangla to English bilingual dictionary. Though some digital English to Bangla dictionaries are available, Bangla to English digital versions are yet to be made available. Moreover, as we are dealing with transliterated Bangla texts, we need a bilingual dictionary that is able to handle the transliterated Bangla (pseudo-English). As none of these are available yet, a cheap mechanism is required for our limited domain (medical) QA system prototype.

A table look-up approach can be implemented to translate the rest of the transliterated Bangla question. The table look-up approach is similar to the QATAR component of the Joost system of Bouma et al. [2005] but, instead of mapping a question to an answer, our system would map parts of the transliterated Bangla question to its equivalent English version. Once such a mapping is obtained for the rest of the question, and the named entities are translated using the two-level transliteration approach, it is possible to obtain a complete English question for further processing.

So, our system would take in “kansar ki?” as input and produce its equivalent English translation which is “What is Cancer?” using a table look-up method as well as translation based on transliteration. Then it will use that English question to do the Question Answering (QA) part.

In the sections ahead we discuss each of the proposed approaches in further detail.

3.3.1 Analysis of *Bangla* Question Structure

One of the reasons behind choosing the medical domain was to make medical information available to the native speakers of Bangla. There are many medical articles and books available in Bangla, too, but only a handful can be found in digital form. But a significant amount of medical information in English is accessible digitally over the Internet. There are many printed medical FAQs available in Bangla but to date no digital versions are publicly available. Some such Bangla printed FAQ's are obtained to study the structure of the questions. Some of the most probable questions are of the following types.

Bangla Question (Transliterated form)	English Question
kansar ki? *(gloss: Cancer what)*	*What is Cancer?*
kansar hole ki korte hobe? *(gloss: Cancer have what to do)*	*What do you do when you have Cancer?*
kanser ki protirod kora jai? *(gloss: Cancer can prevented be)*	*Can Cancer be prevented?*
kansar kivabe choray? *(gloss: Cancer how spread)*	*How does Cancer spread?*
kansar kivabe chikitsha korte hoi? *(gloss: Cancer how treatment to do)*	*How do you to treat Cancer?*
kansar kivabe protirod kora jai? *(gloss: Cancer how prevent do)*	*How can Cancer be prevented?*
bard phlu kivabe chinnito kora jai? *(gloss: Bird Flu how recognize)*	*How do you recognize Bird Flu?*
komon warts kothay hoy? *(gloss: Common warts where occur)*	*Where do common warts occur?*
pregnansi kokhon hoy? *(gloss: Pregnancy when does occur)*	*When does pregnancy occur?*

Table 15: Bangla English Question comparison

In Table 15 some of the simplest common questions in the medical domain are presented. The table gives a complete Bangla question in a transliterated form and its equivalent English versions. Here we notice that for a Bangla question, the disease name (the named entity) is always the first item of the question usually followed by the question element and then any additional verbs. This structure can be found not only for medical questions but for most questions in Bangla.

e.g

Bangla:	*kompiuTar ki?*
English:	*What is Computer?*
Bangla:	*bangladesh'er rajdhi ki?*
English:	*What is the capital of Bangladesh?*

Studying the structure of the questions listed earlier, if a Bangla question is tokenized, then the first token or the first few tokens will be a named entity or more specifically the subject matter of the question. This phenomenon is widely noticeable, specially in the medical domain.

Thus, an approach to translate the Bangla question could be

- tokenizing the transliterated version of the Bangla question,
- using translation based on transliteration to translate the named entities (medical terms)
- translating the rest of the question by a simple table look-up method (*This is definitely not a very ideal approach for a large-scale implementation and we will discuss the issues, however, the prototype system is built upon this simple approach).*

3.3.2 Tokenizing the Question

This is a very trivial issue. As we have already noticed from previous examples in Bangla that each word is separated by a space, same as in English. Thus, a word or a token is

- a set of characters between the start of the question string and the first space
- a set of characters between spaces on either side
- a set of characters with a preceding space and a question mark
- a set of characters with a preceding space and a punctuation mark

Once a transliterated Bangla question has been tokenized, according to our analysis the first or the first few tokens are the named entity or the medical term. Thus the first few tokens are individually processed until a question term is obtained. Once we identify a question term, the tokens before the

first question term are considered the named entity and are translated using transliteration. The rest of the question is translated using the table look-up method for a longest possible match.

3.3.3 Named Entity Translation

We have learned this far that if a named entity is present in a question then it is of utmost importance and describes the subject matter of the entire question. For our medical domain scenario, the disease names or any medical terms are the named entities. And we have already shown that almost all of such medical terms used in Bangla are phonetically similar to English. Thus, it is just a matter of transforming the transliterated Bangla terms into equivalent English version. This can be achieved by a comprehensive rewrite transducer.

We have obtained a list of commonly used medical terms, specifically disease names, from a medical book to further analyze our hypothesis and design the transducer. The list contained 348 disease names in English that commonly occur in humans. Not all of these names/terms were limited to a single word but a good number of these diseases had multiple words like "Dengue Fever", "Yeast Infection" or "Angular Cheilitis". Thus, these multiple word terms were broken down into single words. That resulted in 851 medical terms from those 348 disease names. We organized the terms alphabetically and found that there were multiple entires for a single term at several instances. These duplicate terms resulted from the disease names such as "Atopic Dermatitis" and "Contact Dermatitis". Here we notice that the word *"Dermatitis"* occurs for both the disease names and when they were broken to single terms, 4 medical terms were obtained "Atopic", "Dermatitis", "Contact" and "Dermatitis" with the term "Dermatitis" occurring twice in the list. There is no use of this second instance of the word "Dermatitis" and thus it can be filtered out. Such duplicate terms were noticeable many times in the 851 single-word list. After filtering out duplicate terms there were about 430 different medical terms from the original list of 348 multiple-word disease names.

Those 430 terms were provided to 4 native speakers of Bangla (including 2 medical doctors) to get the transliterated form of the terms roughly following the mapping scheme of UzZaman et al. [2006]. The volunteers were briefed with the phonetic mapping scheme to give an idea in case they were unaware of transliterated Bangla. Then they were asked to provide a transliterated Bangla version of the English medical terms. They were allowed to provide multiple transliterated versions for a single English term but they were not allowed to skip any single English term.

Thus a list like Table 16 was obtained from each of the native speakers of Bangla.

Transliterated Bangla	Actual English
kansar	*cancer*
blad	*blood*
konjunktivitis	*conjunctivitis*
fivar	*fever*
sist	*cyst*
flu	*flu*

Table 16: Transliteration Example

When each of those individual transliteration tables from the native speakers were merged to have a single table, the results showed that there were many transliterated forms for a single English term. This was obvious because Bangla has 50 individual graphemes, each having a different sound, and when these 50 graphemes are mapped to the 26 letters of English, multiple Bangla graphemes are mapped to a single English letter considering the closest pronunciation. Thus when a native speaker tries to transliterate a Bangla term, he or she takes note of the pronunciation as well as the subtle mapping of the letters between the languages and also the actual spelling in the original script. This thinking leads to multiple transliterated versions of a single English term.

Thus, after merging the different versions obtained from the volunteers of this project, there were 796 transliterated Bangla terms against the 430 English terms provided. The final table looked somewhat like Table 17.

Transliterated Bangla	Actual English
kansar *kensar* *kanser*	*cancer*
blad *blud*	*blood*
konjunktivitis *konjancteevytis*	*conjunctivitis*
fivar *feebhar* *phivar*	*fever*
sist *seest*	*cyst*
flu *flue* *phlu*	*flu*
akni *akny* *ekny*	*acne*
dybatis *dybatees* *dibatis*	*diabetes*

Table 17: Transliteration Example

The original list of 430 English terms were randomly separated into two groups of 215 terms each. One group was chosen as the training set and the other for further testing and evaluation. The training set had 215 English terms and their corresponding 279 transliterated Bangla terms. The test set had 215 English terms and their corresponding 517 transliterated Bangla terms. In total, there were 430 English terms and their corresponding 796 transliterated Bangla terms involved in the project.

The training set of 215 English terms and their corresponding 279 transliterated Bangla terms were carefully analyzed and studied. Each of the English terms were broken into individual syllables and so were their corresponding Bangla transliterated versions.

English	Transliterated Bangla
cancer: *can + cer*	*kansar:* *kan + sar* *kensar:* *ken + sar* *kanser:* *kan + ser*
fever: *fe + ver*	*fivar:* *fi + var* *feebhar:* *fee + bhar* *phivar:* *phi + var* *fiver:* *fi + ver*
ulcer: *ul + cer*	*alsar:* *al + sar* *alser:* *al + ser* *aalcer:* *aal + cer* *aalser:* *aal + ser* *aalsar:* *aal + sar*

Table 18: Syllabified terms

The syllabified versions of all the terms were obtained by hand, using careful inspection. Then a table was prepared to compare the syllabified English with the corresponding syllabified transliterated Bangla versions.

English	Transliterated Bangla
can	*kan* *ken* *kan*
cer	*sar* *cer* *ser*
fe	*fi* *fee* *phi* *fi*
ver	*ver* *bhar* *var*
ul	*al* *aal*

Table 19: Mapped syllables

After obtaining a mapping of the syllables, each of those syllables is further broken to analyze the smallest possible character correspondence between English and the transliterated Bangla.

English	Transliterated Bangla
c + a +n	*k + a +n* *k + e + n* *k + a + n*
c + e + r	*s + a + r* *c + e + r* *s + e + r*
f + e	*f +i* *f + ee* *ph + i* *f +i*
v + e + r	*v + e + r* *bh + a + r* *v + a +r*
u + l	*a + l* *aa + l*

Table 20: Character-level mapping

After analyzing those character-level correspondence between the syllables of the English and the pseudo-English (transliterated Bangla) we managed to obtain a pattern. The pattern is very similar

to the phonetic mapping scheme of UzZaman et al. [2006] for English and transliterated Bangla. Thus, from those 215 English terms concerned, about 100 mapping rules were initially obtained. An extract of the mapping rules is shown in Table 21.

Bangla	English
aab	*ab*
ak	*ec*
ak	*ac*
al	*ul*
al	*wal*
ba	*bi*
bag	*bug*
char	*ture*
kri	*cry*
poks	*pox*

Table 21: Mapping Rules

These rules are responsible for replacing the longest possible character sequence of the pseudo-English version to the corresponding English version. These mapping rules, they were implemented as a rewrite transducer. A run was conducted after implementing the transducer with all the obtained rules. We noticed in our first run that each pseudo-English input (transliterated Bangla term) produced an average of 1.7 outputs (actual English name). Of all the produced outputs, only 12% were found to be correct English spellings.

Then by analyzing the produced outputs, a few more mapping rules were identified that were missing in the first run. Also, by accommodating some further longest matches of the letter sequences and slightly modifying the existing rules, an improvement in performance was noticed. With these new modifications and addition of new mapping rules, the transducer produced about 37% correct English spellings; however, this lead to more outputs generated per input. The modification led to generating an average of about 5.7 outputs per input, with the worst case of a single input generating over 23 outputs. The process of obtaining the rules is not done using any standard machine learning setup. The rules are all handcrafted with the aim that, with a limited generic mapping rules, a good number of correct transducer outputs will be generated. With that aim in mind the dataset was purposefully chosen the way was mentioned.

Though the modifications helped in the improvement of the performance it led to another problem of identifying the correct English spelling or the closest one of all the produced outputs. So a ranking mechanism was essential to rank the most correct English disease name or the medical term. For this implementation, a simple edit-distance count was used to rank the generated outputs. *This was achieved by a rather naive implementation as the domain of words involved was very small. For a large-scale implementation, this ranking mechanism would not be at all efficient.* As there were only 430 English medical terms involved, they were sorted alphabetically and stored on individual files according to the starting alphabet of the term itself. Each of the generated outputs was compared in one of those files and the edit-distance measure was noted. The output that matched with one of the terms in the files had an edit-distance of zero and was most likely the term we were looking for. If none of the outputs had an edit-distance count of zero then the output with the minimum count was taken for further processing. In case two or more terms ended up with the same count, the one on top, after sorting alphabetically, was taken. An example is shown below.

Actual Medical Term: *excoriee*
Transliterated Bangla Input: *ekshori*

Bangla Input	Outputs Generated
ekshori	*acschore*
	acschoriee
	acscore
	acscoriee
	ecschore
	ecschoriee
	ecscore
	ecscoriee
	exchore
	exchoriee
	excore
	excoriee

Table 22: Transducer Outputs

Here we notice that the input generated 12 outputs, and one of the output form is the term we were

looking for. From the outputs generated, the system realizes that the outputs either start with an *'a'* or an *'e'*. So it looks in those two respective files and runs the edit-distance algorithm for each of the outputs. The files involved are shown in Table 23.

Terms starting with 'a'	Terms starting with 'e'
abrasion	*eczema*
abscess	*electrodesiccation*
acanthosis	*epidermoid*
acne	*erosion*
acrochordon	*eruption*
actinic	*erythema*
acuminata	*erythematosus*
acuminatum	*erythrasma*
aid	*ethnic*
alba	*examination*
allergic	*exanthem*
allergies	*excoriee*
alopecia	*exhaustion*
anesthesia	*eye*
angioma	
angular	
animal	
annulare	
anthrax	
aphthous	
areata	
arthropod	
athlete's	
atopic	
atypical	
avian	

Table 23: Medical Terms list

As can be seen for this case, only one output will exactly match one of the terms in the two files and

that is the term we are looking for. As already said this is not a very good implementation to rank the outputs because of the number of comparisons involved. A few hard-coded rules are also implemented to eliminate the most obvious wrong outputs. For example,

Bangla Input	Outputs Generated
fut	*foot* *phoot* *ffoote* *phoote*

Table 24: Outputs Generated

For the input "fut" the correct output is "foot". The other outputs were generated because of the mapping rules such as "f->ph", "f->ff" and "t->te". All these mapping rules were needed to accommodate other terms, for example, for the input "dandraf" the output should be "dandruff" and so the mapping rule "f->ff" is needed. But in English no word starts with "ff" so for the input "fut" the output "ffoote" can easily be discarded as it violates a basic language model rule. Having such rules help to eliminate some obvious wrong outputs.

3.3.4 Table Look-Up Translation

Once the medical terms are translated it is the turn to translate the other words within the Bangla question. As this study involves a very limited/closed domain, the most common medical questions were analyzed from different medical FAQs available both for English and Bangla. A list of those simplest medical questions in Bangla was prepared and an equivalent English translation was also prepared. Thus a final list of just about 20 question variations were prepared to be considered in this prototype. The current implementation searches for the longest match of words (the Bangla question terms) in a file which also contains their equivalent English versions.

Rest of the Question (Bangla:Transliterated form)	Equivalent English
ki? (gloss: what)	*What is -?*
hole ki korte hobe? (gloss: have what to do)	*What to do when you have -?*
ki protirod kora jai? (gloss: can prevented be)	*Can - be prevented?*
kivabe choray? (gloss: how spread)	*How does - spread?*
kivabe chikitsha korte hoi? (gloss: how treatment to do)	*How do you treat -?*
kivabe protirod kora jai? (gloss: how prevent do)	*How can - be prevented?*
kivabe chinnito kora jai? (gloss: how recognize)	*How do you recognize -?*
kothay hoy? (gloss: where occur)	*Where does - occur?*
kokhon hoi? (gloss: when does occur)	*When does - occur?*

Table 25: Rest of the Question Translation

The equivalent English version is the rest of the question that is required along with the medical term to generate the English question. Further discussion and results can be found in the next chapter.

3.3.5 English Question Generation

To generate a correct natural language sentence computationally, a POS tagger and a syntactic parser plays a major role. In this case no such tools are yet freely available for Bangla. Thus, a quick and simple workaround has been implemented for this limited domain framework design.

For almost all the medical questions in Bangla, the question starts with the disease name itself followed by the question elements and any other verbs. Table 26 illustrates the phenomenon.

Bangla Question (Transliterated form)	Equivalent English Question
kansar ki? (gloss: Cancer what)	*What is Cancer?*
kansar hole ki korte hobe? (gloss: Cancer have what to do)	*What do you do when you have Cancer?*
kanser ki protirod kora jai? (gloss: Cancer can prevented be)	*Can Cancer be prevented?*
kansar kivabe choray? (gloss: Cancer how spread)	*How does Cancer spread?*
kansar kivabe chikitsha korte hoi? (gloss: Cancer how treatment to do)	*How do you treat Cancer?*

Table 26: Bangla English Question comparison

Thus, the named entity Translation part considers the first or the first few tokens of the input string and uses the transducer to translate the disease name to its correct English version. The rest of the input string is searched for a longest match through the table look-up translation method. Once a match is found from the table look-up translation, the blank space reserved for the named entity (in this case the disease name or the medical term) is replaced with the output produced from the named entity Translation part. Again, this is not a very elegant solution but in this limited experimental scenario, where the performance in terms of speed is not taken into consideration, these naive approaches are enough to prove the hypothesis of the overall project.

3.4 Implementation Decisions

We list here the tools used to implement the prototype.

3.4.1 FSA Utilities Toolbox

The highlight of the prototype framework is the *translation based on transliteration* part which is used to translate the named entities (the medical terms) of a transliterated Bangla question (pseudo-English) to its actual English versions. The syllabified mapping rules obtained in the *Design* section can be implemented in many ways. Each of the mapping rules can be hard-coded with a programming language of choice and used in the implementation. The other way to implement the mapping rules is to build a rewrite transducer using the rules. The transducer is responsible to read a character sequence which in this case is a medical term written in pseudo-English. The transducer compares each character or a sequence of characters within the input term with the available rewrite rules. If a rewrite rule is available for a character or a character sequence then the corresponding output of the rule is written in that particular position of the input term. If no rewrite rule is present

for a character then it is copied as it is to the output. The rewrite transducer for the Named Entity Translation part has been prepared using the FSA Utilities Toolbox. FSA6.2[40] is available under the GNU general public licence. It is a collection of utilities to manipulate regular expressions, finite-state automata and finite state transducers.

3.4.2 Python

Python is a very powerful dynamic programming language which is used in a variety of application domains. Python's vast standard library and flexible coding style makes it a very popular and efficient programming language to be used in Natural Language Processing applications. The ranking of the outputs from the Named Entity Translation, Table look-up Translation and the English Question Generation part is prepared using Python 2.5.2.

3.4.3 JavaServer Pages

The web-interface of the prototype framework has been prepared using JavaServer Pages (JSP). JSP is platform-independent technology that allows rapid development and easily maintainable dynamic webpages.

3.4.4 Apache Tomcat Server

The open source Apache Tomcat server was used to handle the JSP technologies involved. Apache Tomcat is developed by the Apache Software Foundation (ASF) and provides a HTTP server environment for Java code to run.

3.5 Program Flow

The web-interface takes in a Bangla question in a transliterated form, calls in a Python script to tokenize the question, then FSA is called to translate only the named entity within the question. The output from the transducer is saved in a file. A Python script uses the transducer output and the result of the table look-up process to generate the English version of the Bangla Question. Once the English question is ready, Google is passed with the exact question to obtain the answers. In this implementation the results from Google are just studied but not further processed for actual answer generation.

40 Available at http://www.let.rug.nl/~vannoord/Fsa/fsa.html

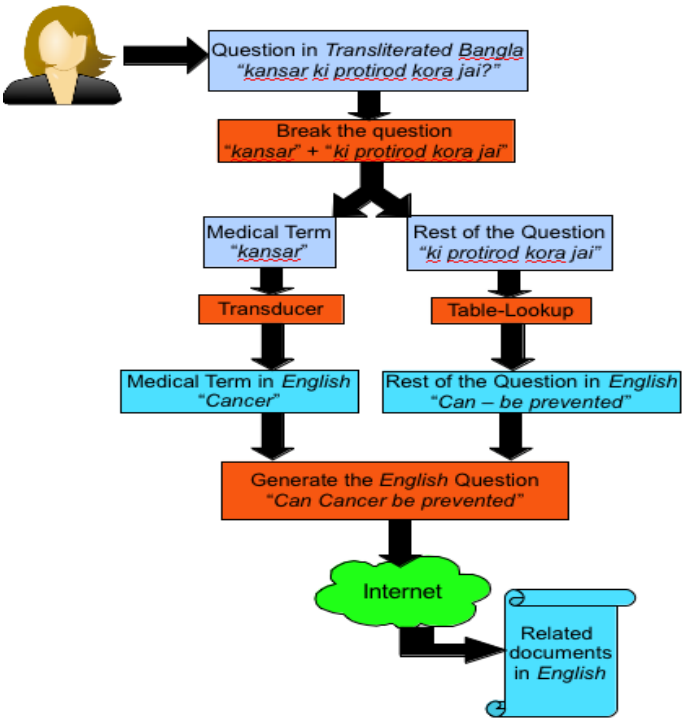

Fig 18: Architectural Diagram

3.6 Summary

This far we have introduced our aims of the project and have given the detailed steps in designing the prototype framework. We have repeatedly mentioned at several places that the aim of this project is not to build a complete Bangla question answering system but to propose an approach to solve a subset of a complete Bangla question answering task, thus the prototype deals with very limited cases to prove the hypothesis of *translation based on transliteration and table look-up method*. Some of the proposed ways such as the table look-up and the ranking method of the medical terms would not be a good approach in large-scale implementations as in such large-scale implementations the number of question types and the total number of medical terms involved would be much larger. And with the increase in number of types and terms, the number of comparisons in the table look-up approach would increase exponentially leading to the overall poor performance of the system in terms of speed. The *Future Work* section presents those issues and their possible solutions. In Chapter 4 we evaluate our system and discuss the results.

4 Analyses, Evaluation and Discussion

In this chapter we describe the overall performance and the limitations of the implemented system. We present some results for the individual components involved in the system. Then we evaluate the entire implementation as a whole. We also present some comparisons with similar other implementations at appropriate sections of this chapter as well as the next chapter.

4.1 Translation task

In a cross-language QA scenario there are two main tasks. The translation task and then the QA task. The QA task's performance is highly dependent on the translation task as the translated output will be used for the actual question answering. If the quality of the translation is not good then no matter how good the QA component is, it is bound to give bad results. The quality of a machine translated text are mostly evaluated using the BLEU[41], NIST[42] and METEOR[43] scores. All these metrics are very suitable for larger datasets but the translation component of this project deals with a limited set of text that requires translation so the conventional metrics measures available to evaluate translations is not applicable here. We have formulated our own evaluation mechanism to evaluate individual components of our implementation. As a simple implementation strategy has been employed in the overall design, each of the components are tested empirically with our controlled dataset.

4.1.1 Transducer Outputs

We have used rewrite transducers to obtain the translation of the medical terms. We had access to 348 disease names in English and our implementation is based on these terms. Of the 348 disease names there were 430 different single-word terms in English. We randomly separated these 430 single-word terms into two equal groups (215 terms on each group). One group was used to build the mapping rules and the other group was used later to evaluate the transducer output for unknown inputs.

41 Bilingual Evaluation Understudy – It is a measure to evaluate the quality of a machine translated text while comparing with a version translated by human judges. The score accounts for adequacy by looking at word precision and accounts for fluency by calculating n-gram precisions. Also a brevity penalty is there to compensate for recall. The final score is calculated by a weighted geometric average of the n-gram scores over a large set of test data [Papineni et al 2001].

42 It is another metric to evaluate the quality of a machine translated text and is based on the BLEU metric, however, it takes the arithmetic mean of the n-gram counts unlike the geometric mean in BLEU metric [Doddington 2002].

43 Metric for Evaluation of Translation with Explicit Ordering – It gives a score based on explicit word to word matches between the translation and a given reference [Agarwal et al. 2008].

We used the 215 English terms of the training set to build the initial 100 mapping rules for the transducer. Those 215 English terms had 279 corresponding pseudo-English (transliterated Bangla) terms. Once these 100 mapping rules were implemented, the same 279 pseudo-English terms of the training set were used to check the performance of the transducer.

Test Set Size	# of mapping rules	Total # of outputs	Average output per input	# of correct terms	% of correct outputs
215 *English* terms, 279 *pseudo-English* terms (training set)	100	475	1.7	33	11.8

Table 27: Test Run 1

After our initial run on the same training set the percentage of correct output was just about 12%. We evaluated each of the 475 outputs against the 215 terms and found many mapping rules to be missing. We added 17 more rules and modified some of the existing ones. We have already mentioned earlier that this implementation does not employ any machine learning techniques. The rules are all handcrafted and we expect to translate as many correct terms possible with a generic set of rules. Thus the smaller dataset (215 English and 279 pseudo-English terms) was deliberately chosen to design the mapping rules so that they can be tested on the larger dataset (215 English and 517 pseudo-English terms).

Test Set Size	# of mapping rules	Total # of outputs	Average output per input	# of correct terms	% of correct outputs
215 *English* terms, 279 *pseudo-English* terms (training set)	117	642	2.3	41	14.6

Table 28: Test Run 2

After the modification and addition of few rules we noticed that the percentage of correct outputs improved by about 3% but with that the total number of outputs generated by the transducer increased significantly producing about 2.3 outputs per input.

Test Set Size	# of mapping rules	Total # of outputs	Average output per input	# of correct terms	% of correct outputs
215 *English* terms, 279 *pseudo-English* terms (training set)	124	1004	3.6	61	21.8

Table 29: Test Run 3

As with some modifications and addition of rules an improvement was noticed, so the output was further comprehensively analyzed to identify any missing rules. Our initial mapping rules were mostly limited to one or two character sequences such as "a->e", "k->c" or "aa->a". These smaller sequences were contributing mostly in generating more outputs without improving the overall performance. Thus some of the longest character sequences were obtained like "char->ture", "shori->coriee" etc. With about 6 such new rules and other additions the performance was further evaluated (Test Run 3). And this time we noticed a significant improvement over our previous runs but with that the average number of outputs also increased. This means that the recall of the system was going down. After further adjustments to the rules we ended up with a final 129 mapping rules. We used the test set to evaluate the transducer. The test set had 215 English terms with their corresponding 517 pseudo-English terms.

Test Set Size	# of mapping rules	Total # of outputs	Average output per input	# of correct terms	% of correct outputs
215 *English* terms, 517 *pseudo-English* terms (test set)	129	2430	4.7	212	41

Table 30: Test Run 4

With the test set of 517 pseudo-English terms the transducer strangely produced about 41% correct outputs. We analyzed our data further to find a reason behind such improvement and we noticed that the test set had more terms where the smaller mapping rules ("*i->i*", "*i->e*", "*e->e*", "*e->a*" etc.) were used. This lead to the generation of more outputs but overall it was producing correct outputs most of the time. With these results we made another final run with all the 796 pseudo-English

terms to evaluate the transducer.

Test Set Size	# of mapping rules	Total # of outputs	Average output per input	# of correct terms	% of correct outputs
215 *English* terms, 796 *pseudo-English* terms (test set)	129	4139	5.2	295	37

Table 31: Test Run 5

With all the 796 pseudo-English terms the transducer produced about 37% correct outputs. The transducer also generated an average of about 5.2 outputs per input term. We also noticed that in the worst case scenario the transducer produced 39 outputs for a single input term.

4.1.2 Table look-up Approach

For the translation of the rest of the question a cheap table look-up mechanism was employed. The implementation looks for the longest possible word sequence from the input string. This is not a very ideal and elegant solution because in a larger implementation scenario there will be hundreds of question variations. Thus the number of comparisons will multiply with the number of question types addressed. This will lead to a slower performance of the overall QA task. In our limited implementation we dealt with 20 different questions types only that were obtained from the medical FAQs and the medical resources used in the project and were verified by the volunteers of the project.

Bangla Question (Transliterated form)	English Question
kansar ki? *(gloss: Cancer what)*	*What is Cancer?*
kansar hole ki korte hobe? *(gloss: Cancer have what to do)*	*What do you do when you have Cancer?*
kanser ki protirod kora jai? *(gloss: Cancer can prevented be)*	*Can Cancer be prevented?*
kansar kivabe choray? *(gloss: Cancer how spread)*	*How does Cancer spread?*
kansar kivabe chikitsha korte hoi? *(gloss: Cancer how treatment to do)*	*How do you treat Cancer?*

Table 32: Bangla Questions

In Table 32 we see some typical questions asked in Bangla and their corresponding English versions. From such a list of our 20 question variations we excluded the medical term itself and produced a table. An extract shown in Table 33.

Rest of the Bangla Question (Transliterated form)	English Question
- ki? *(gloss: - what)*	*What is -?*
- hole ki korte hobe? *(gloss: - have what to do)*	*What do you do when you have -?*
- ki protirod kora jai? *(gloss: - can prevented be)*	*Can - be prevented?*
- kivabe choray? *(gloss: - how spread)*	*How does - spread?*
- kivabe chikitsha korte hoi? *(gloss: - how treatment to do)*	*How do you treat -?*

Table 33: Rest of the Question

From Table 33 we generated a list of stop words which will help the system to identify the end of a medical term in the question and the start of the rest of the question. So for Table 33 the list of stop words would be

“ki” , “hole” and “kivabe”

The implementation looks for one of these terms from the beginning of the entered question (in pseudo-English), and as soon as it finds one, the word or words before that stop word is the medical term to be translated using the transliteration mechanism and the rest of the words in the question starting and including the stop word itself is searched in a file implemented like Table 33. The implementation looks for an exact string match entry for the rest of the question words that were entered. Once an entry in the table matches, the corresponding English version of the rest of the question is taken for processing. The obtained English version has a marker within the question (in our case we had a hyphen) which is replaced by the term or the terms obtained from the transliteration mechanism. Following are the results of some test runs.

Input	Output	Result	Observation
kansar ki (gloss: kansar what)	`what is cancer`	Correct	The medical term was translated correctly and there was a correct table look-up entry. The generated question is a correct generation.
ekny hole ki korte hobe? (gloss: ekny have what to do)	`what to do when you have acne`	Correct	The medical term was translated correctly and there was a correct table look-up entry. The generated question is a correct generation.
folikulytees ki (gloss: folikulytees what)	`what is faleeacaleiaetic`	table look-up is correct but transliteration mechanism gave wrong output	Here the medical term was one of those that didn't produce a correct term. The top ranked term is not very close to the actual term "*Folliculitis*"
melanositik nevas kokhon hoi? (gloss: melanositik nevas when occurs)	`when does maleanuciaetic nevus occur`	table look-up is correct but transliteration mechanism gave one correct and one wrong output	Here there were two medical terms involved. One was translated correctly but the other one was somewhat close enough but not the correct one.
komon and klasikal migrane kivabe chikitsha korte hoi? (gloss: komon and klasikal how treatment to do)	`how to treat common and classical migreyn`	table look-up is correct but transliteration mechanism gave partially correct output.	This is an interesting case. Here the medical term involves more than one word and one of them is actually a conjunction. The conjunction "*and*" has a literal translation but this phenomenon was tested the volunteers and all of them kept the original one. Now the system expects pseudo-*English* term to be translated but "*and*" was a correct *English* term and because of some mapping rules the "*and*" was translated correctly to "*and*" itself.

Table 34: Question Generation

Table 34 shows how the transliteration mechanism and the table look-up method together

performed to generate the English question. The quality of the generated question is very hard to evaluate in our implementation scenario because a major portion of the generated question is done by looking it up in a table containing correct translations. So the question part without the medical terms should always turn out to be a correct translation for our implementation. However, in our tests, in only 72% of the cases the rest of the question part was found using our table look-up approach. Though we had a very limited question variation set (20 only) and we limited our evaluation to only those variations, those variations could be spelled in more than one way just like the medical terms in pseudo-English (transliterated Bangla) so if the entered question in pseudo-English was not an exact match in our table look-up, the system behaved erratically or did not produce a question at all. This phenomenon of the system is explained in the next section. Whenever the translation mechanism produces a correct medical term along with a correct table look-up entry, the generated question is definitely a well-formed question. In our testing 53% turned out to be a well-formed question without any spelling mistakes. We further evaluated those well-formed questions and found that 83% of those had only a single word medical term. We tried to analyze a bit further and noticed that most of those medical terms accommodated the longest character sequence mapping rules. We tested our well-formed as well as not so well-formed questions using Google and the observations are stated in section 4.2 of this chapter.

4.1.3 Exceptions, Assumptions and Limitations

The entire prototype has been designed and implemented considering some assumptions and exceptions. The main reason behind most of the assumptions and exceptions is due to the lack of language specific resources and tools. And it was beyond the scope of this thesis to properly address those exceptions before, and then design and build the proposed framework. However, the main objective of this study was to propose something effective within such limitations. Setting aside the greater limitation such as the lack of resources and the time to build them, we had to further accommodate some limitations in our design. They are listed below:

1. Not all the medical terms in Bangla are imported words. There are native Bangla translations available for some medical terms such as “Heart” and “Stomach” are “hridoy” and “pakostholi” in Bangla but their imported versions are equally used. So if a user preferred to use the actual Bangla word instead of their imported forms the system would behave erratically as the system is in no way capable of identifying whether the entered text was Bangla, transliterated Bangla or English. And as this implementation does not make use of a Bangla to English dictionary, only the imported versions have been considered.
2. There could be many transliterated Bangla versions for a single term if the phonetic

mapping rules of UzZaman et al. [2006] are comprehensively used. However, in this implementation only the most common forms have been considered that are enough to phonetically represent the English terms. The term “stomach” can be written as “estomak”, “stomak”, “estomach”, “estomuk”, “stomuk” etc. Though we have considered many transliterated versions of an English term, we surely have not considered all the possible transliterated versions of the medical terms. We considered only those that were obtained by consulting native speakers of Bangla and the volunteers of this project.

3. Only the simplest type of medical questions are being handled in this prototype implementation. Questions like “How can babies be infected by Chicken Pox?” in Bangla is “bacchara ki chiken poks-e akranto hote pare” or “Can Avian influenza affect both adults and children?” in Bangla is “bacchara ebong boro-ra, duijon-i ki ebhian inphluenza dara akranto hote pare” have a far more complex structure in Bangla than the ones handled in this implementation. These types of questions may not be generalized by the structure “disease name followed by the rest of the question”. A good syntactic as well as dependency parser is essential to generate such questions. We have already learned how different types of parsing can help in the overall QA task in many ways. As our implementation scenario is proposed for an open text collection such as the Internet, the parsing technologies cannot be used exhaustively to parse all the text available over the Internet but it could definitely be used to parse a pseudo-English version of a question and then generate the actual English question. Further, in Bangla the medical terms may include case markings in complex question cases like the ones just stated above. Thus a morphological analysis is also essential to obtain the correct medical term out of the entire question. Generating a good quality question without a table look-up method that we proposed is a major research area of its own.
4. Only one transliterated form of the rest of the question is considered for our implementation; however, each word used in the table look-up translation can have more than one transliterated form.

   e.g.

English Word	Bangla Transliteration Used	Other Possible Transliteration
what	*ki*	*kee*
how	*kivabe*	*kibhabe, keevabe*
prevent	*protirod*	*protirodh, proteerodh*
treat	*chikitsha*	*chikitsa, cheekitsa, cheekeetsha*

Table 35: Other Possible Transliterations

In our implementation we had only one version of these words (the words occurring in the rest of the question). We considered only those versions that were the most probable and easier to spell, keeping in mind the mapping scheme of UzZaman [2005]. We verified our version with our volunteers, too.

4.2 Question Answering Task

The QA part of this thesis was of secondary importance in the entire project and thus most of the analyses presented here are empirically tested. We proposed a method where a user types in a question in a transliterated form and gets answers from a system which does not necessarily work with the user's native language. *So we basically proposed a translation mechanism as an interface for question answering.*

From our implementation we managed to obtain 53% well-formed translated questions. We tested these well-formed as well as not so well-formed translated questions with Google to understand their behavior. Google is a very popular Internet search engine. It started as a basic information retrieval engine but over the years its has adopted many techniques and heuristics within its searching mechanism that it is now far more than just a keyword based information retrieval system. We cannot claim that Google is now a complete question answering system but it does perform well with some question types though it does not actually produce a complete answer. Here we show some tests that we performed.

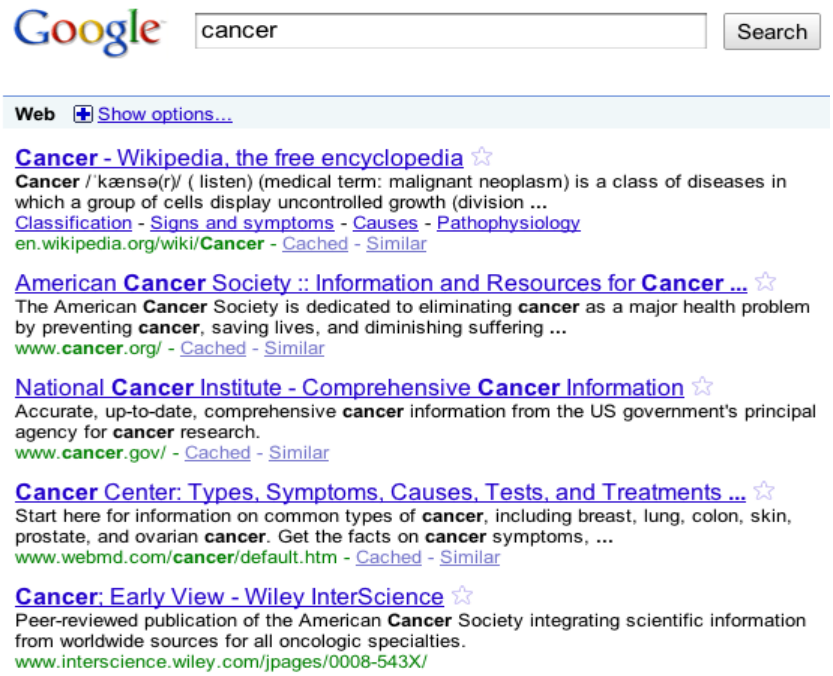

Fig 19: Google search with "cancer"

In the above figure we searched for the term “cancer” and we noticed that the top 5 results are related to our search key. Google even highlights our search key within the topic and the snippets it presented. Now we use a complete question which includes the same search key.

Fig 20: Google search with "what is cancer"

Here we notice that Google has used all the terms in the search query. And the snippets and most of the topics include the entire question we asked. Google appears to use the rule[44] that if a search query starts with “what” then it manages to find the topic of interest within the query and looks for a web definition for that topic and presents it as the first result. The snippet of that definition is in most cases the actual answer to a query. However, in the above case, we did obtain a definition and it is correct but not relevant to our query as we meant “cancer” in the medical sense. Thus even though our implemented system was able to produce “what is cancer” correctly we cannot limit our answer to the web definitions provided. The definition provided is completely wrong for the medical domain. Thus we need to further look into the results for an answer. However this was a very special case as the term “cancer” has more than one meaning. But if we repeated our test with a different term the web definition itself is enough as an answer to our question as can be seen below.

44 Google does not disclose their internal techniques and algorithms. The observations stated here are found by performing many searches through Google.

Fig 21: Google search for "what is conjunctivitis"

Now we tried the same question with our system. So, we typed in “konjunkteevytis ki” in pseudo-English form and we obtained “what is chonjanctivitis” from our system. This is an example of a not so well-formed output. Our system did not manage to produce a correct spelling for the term “conjunctivitis” but it produced “chonjanctivitis”. We tried the exact output as a Google search query.

Fig 22: Google search for "what is chonjanctivitis"

Here we have given a wrong spelling but even then Google has managed to give the correct results and it also proposes the correct spelling for the medical term. We tested this phenomenon a few more times.

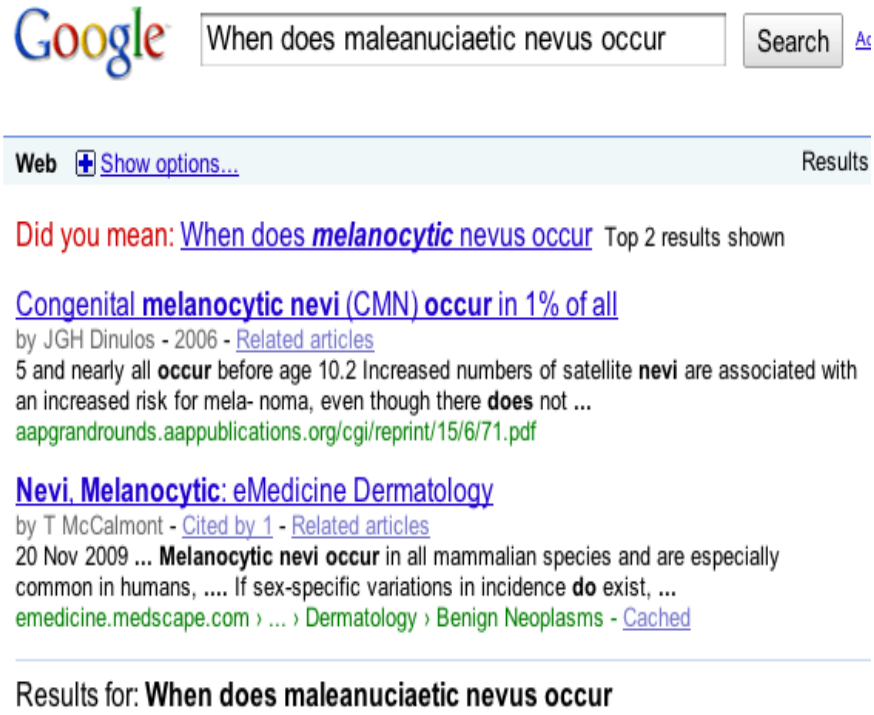

Fig 23: Google search for "when does maleanuciaetic nevus occur"

We typed in “milanosytik nevas kokhon hoi” in pseudo-English form and we obtained “when does

maleanuciaetic nevus occur" from our system. This is another example of a not so well-formed output. Our system did not manage to produce a correct spelling for the term "melanocytic" but it produced "maleanuciaetic". The other medical term was spelled "nevas" in pseudo-English and it produced the correct English spelling "nevus". Google was able to understand our wrong spelling and managed to even suggest the correct one. These observations are very promising for us as we have somewhat managed to ask a question in Bangla (we wrote it in a transliterated form) and we managed to search an English text collection and obtain some relevant results. This definitely proves that our proposed system can play a major role in a cross-language QA task without having major language processing tools for the questioning language (here it is Bangla). However, in our experiment there were certain cases where Google could not produce a result at all. An example is shown below.

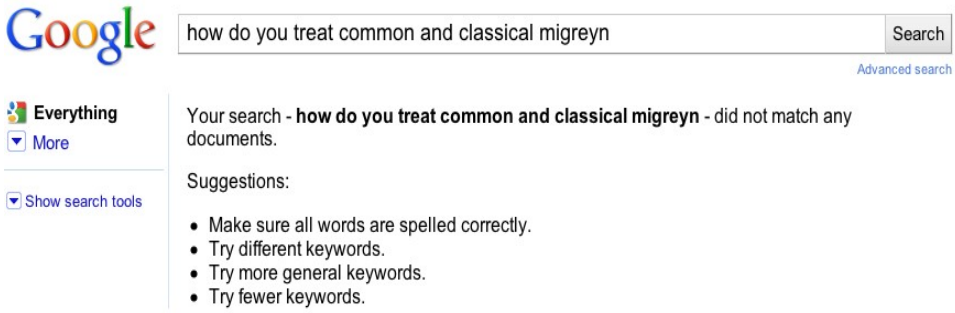

Fig 24: Google search for "how do you treat common and classical migreyn"

We typed in "komon and klasikal mygren kivabe chikitsha korte hoi" in pseudo-English form and we obtained "how to treat common and classical migreyn" from our system. This is another example of a not so well-formed output as one term is not spelled correctly. Our system did not manage to produce a correct spelling for the term "migraine" but it produced "migreyn". The other medical terms were correctly translated. Google was not able to find or suggest any relevant results. We corrected the spelling to see if Google was really able to find some results and Google did manage to find relevant results.

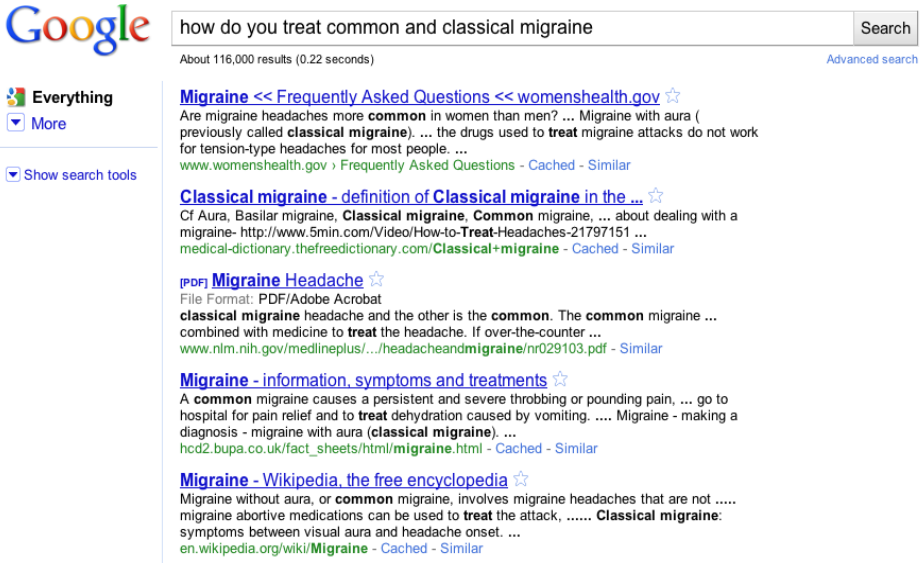

Fig 25: Google search for "how do you treat common and classical migraine"

As we have already mentioned that we have not implemented any techniques to process the output from Google but we have definitely managed to prove our hypothesis of *translation based on transliteration and table look-up as an interface for a cross-language question answering system* in a very controlled environment. If our system provided a well-formed translation then in most of the cases Google provided relevant information as top results unless the medical term involved has multiple meanings as we have seen for the term “cancer”. If the question contains multiple medical terms then there is further high probability of obtaining relevant results from Google even if one of the terms had multiple meanings.

4.3 Overall Analyses and Discussion

This far we have shown how our proposed design and its implementation performs. As the entire design is hard to evaluate with a single metric, we evaluated each component individually with our own methods. The named entity translation part was able to produce an overall 37% correct output from the transducer in our tests. We evaluated the table look-up mechanism individually and found that 72% of the times we managed to obtain a correct entry. This could actually be improved and also the limitations stated earlier could be avoided by allowing the user some sort of templates to choose from rather than typing the question themselves. That will eliminate the different spelling versions involved in the process. We then found that 53% of the questions that we obtained from our volunteers were actually well-formed questions by which we meant that a correct table entry was found and the transliteration mechanism produced a correct output too. We tried most of our well-formed questions with Google and we were able to locate relevant results in most cases. We

also tried some of our not so well-formed questions by which we meant that there were spelling mistakes in the medical terms (produced by the transducer). Google was able to understand where the spelling mistake was and was able to suggest correct spellings as well as relevant results without even modifying the question we obtained from our implementation.

These observations show promising results in a constrained environment. We have managed to use a Bangla question to look for relevant answers from the Internet without having a proper machine translation engine. *We have managed to show that this technique of translation based on transliteration and table look-up can act as an effective interface in a cross-language question answering scenario within a controlled environment.*

5 Future Work and Conclusion

5.1 Conclusion

Through this thesis work, we tried to learn the important issues in the field of Question Answering (QA) systems. We peeked into the internals of many established QA systems. We explored the capabilities of each of them and the reasons that make them good at their task. Then we looked into the details of cross-language QA tasks. We learned that all the systems that do support multiple languages or work in a cross-language environment have access to well-established machine translation systems. Apart from that a good number of them are heavily dependent on pre-processed contents. Once they had a base system they enhanced their base system with several other components and features which significantly improve their QA task.

From our research findings we took the initiative of proposing a basic framework for a QA task for the language Bangla. Bangla is one of the top 10 most widely spoken languages of the world with over 200 millions speakers, however, having such a vast speaker base the language lacks many of the basic language processing resources and tools that are already available for other languages. There are ongoing projects to make those tools and resources available for public use but the entire initiative is behind schedule compared to the language's presence in the world. Many tools and resources have already come out and are maturing day by day. However, there are no known initiatives for a digital Bangla QA system. We have tried to grasp this opportunity and propose a basic QA system for Bangla but we were at an obvious disadvantage in terms of the resources and tools that were required. We learned that to have a complete open-domain Bangla QA system the first thing we need is a significant quantity of digital Bangla text. There are only a handful of Bangla corpora available and most of them are genre specific. Then we needed tools that could process Bangla text and grammar to query the text collections and generate the answers. We learned that some grammar processing tools with very limited capabilities are available for regular Bangla sentences but not for Bangla questions. Then we explored the possibility of a cross-language QA environment where a user would ask a question in Bangla but the system would generate an answer from texts in a language other than Bangla and translate it back to Bangla for the user. This idea is feasible when a robust machine translation engine is available between Bangla and the other language(s) involved in the cross-language QA task. Unfortunately this was not possible either as a mature machine translation engine between English or any other language to Bangla has yet to be

developed. There are some systems available to translate English texts to Bangla but for our proposal we also needed a Bangla to English translation engine to translate the Bangla question to English and then process the question in an English QA system. With all these limitations we narrowed down our initial idea of building a complete open-domain Bangla QA system to building just a small and effective interface for the Bangla QA task in a very controlled environment. According to our proposed framework the interface is able to take in a Bangla question in a transliterated form and query an Internet search engine that works with English texts. We used transliterated Bangla as our input language as transliterated Bangla is very popular in day to day communications and minimizes the issues with Bangla script handling.

We proposed a transliteration and table look-up based implementation as an interface for a digital Bangla QA scenario. We limited our domain to only certain varieties of medical questions. The reason behind choosing the domain was that medical terms in the Bangla language sound pretty close to their English counterparts. So we proposed and finally proved a method to use finite state transducers to translate the medical terms written in transliterated Bangla to their original English spellings. We were able to achieve 37% correct translations for the medical terms which is close to the accuracy of Jiang et al. [2007] who implements a similar strategy to translate named entities and were able to obtain 48% correct translations. We used a very naive table look-up approach to translate the rest of the question and generate the complete English question. People may easily argue that table look-up approach is too simple a technology to be implemented for this task. We agree but given the constraints that we have mentioned time to time in this document, table look-up was the only possible approach to show a working interface. Earlier we didn't have a single way to forward a Bangla question to an English search engine but at least with our simple approach we have been able to ask a question in Bangla and get some results from English documents. With the transliteration module and the table look-up method combined we were able to translate 53% of our questions correctly from Bangla to its equivalent English versions. We observed that our transliteration module despite failing partially or completely in translating the medical terms in certain instances, our overall implementation strategy of the interface managed to guide the search results to the right directions. We learned from Zhang [2004] that the snippets provided by Google are enough to find answers to most of the question. With our proposed interface and some processing of the results that we obtain from Google we can get further closer to generating a correct answer to our question.

We find the results of our implemented system very promising and strongly believe that this strategy can be modified, redesigned and extended to enhance the future of Bangla QA task.

5.2 Future work

We believe that we were able to achieve satisfactory results to prove our hypothesis of a *possibility of translation based on transliteration and table look-up as an interface for a limited domain QA task.*. Our prototype framework can be extended with many other technologies. The assumptions and limitations that we stated in section 4.1.3 might be addressed first in the next version of the system.

The mapping rules in our implementation are all handcrafted. We started with a small set of rules and extended them as we came across new cases. A machine learning approach might be employed in refining and extending the rules. A supervised or adaptive learning strategy can be devised to make the transducer learn new rules. Further, a scoring mechanism for the rules can be employed so that rules that produce the correct translations are preferred over the other rules.

We have mentioned that apart from medical terms there are many other words, too, that are imported into the Bangla language on which we based our hypothesis. The words in fields involving Engineering and legal systems are also mostly loan-words for Bangla. The system can be extended to accommodate those terms, thus extending the implementation beyond the medical domain.

We implemented our system with limited dataset and so we checked for the correct spellings within our dataset itself. Here an English dictionary would be necessary when the system is extended beyond its medical genre.

We have mentioned already that the table look-up approach will not be a very elegant method in a large-scale implementation. Thus, new techniques need to be explored to accommodate more variations in questions. A good bilingual dictionary (Bangla to English) can be used to translate the individual words (transliterated *Bangla to English*) and find their respective POS tags. English grammar rules can be used to generate the English version of the question from the individually

translated words. Until such a bilingual dictionary becomes available a different strategy could be explored from some existing systems. Haque [2006] implemented a system to convert transliterated Bangla (pseudo-English) texts to Unicode encoded Bangla texts. A similar strategy might be deployed in the current implementation to get the actual Bangla version of the rest of the question. We learned that Hossain [2008] developed an open-source English to Bangla MT system which produced Unicode encoded Bangla texts. An attempt could be taken to reverse the procedure of Hossain [2008] so that the Unicode encoded Bangla words can be used to obtain their English counterparts. The process to reverse an English to Bangla MT system is definitely not a straight forward task and might end up as a task of designing a Bangla to English MT system from scratch.

We have used the exact translations that we obtained to search for an answer. We have learned about query expansion methods in our research and that could be explored further and implemented in our framework for better and relevant results. Also the verb forms that are in use in our table look-up method could be changed and expanded for better results in retrieving relevant documents.

During our research we learned about some ongoing projects on English to Bangla machine translation systems. It would definitely be a good idea to interface such a translation engine to accept outputs that Google returns for our Bangla questions. That way the user will actually get the output in Bangla.

We believe that despite having resource limitations and time constraints for this project, we managed to get a step closer to having an a full fledged Bangla Question Answering system.

6 Appendices

Extract of the Mapping Rules

```
%%          -*-Mode: prolog;-*-
:- multifile rx/2.
:- multifile macro/2.
:- discontiguous macro/2.
:- discontiguous rx/2.

macro(bangla_eng,

     replace({
               [a]:[e],
               [a,a]:[a],
               [a,a,b]:[a,b],
               [a,k]:[e,c],
               [a,k]:[a,c],
               [a,l]:[a,l],
               [a,l]:[u,l],
               [a,l]:[w,a,l],
               [b,a]:[b,i],
               [b,a,g]:[b,u,g],
               [b,y]:[b,i],
               [c,e]:[c,e],
               [c,e]:[c,h,e],
               [c,i,s]:[s,c,e,s,s],
               [c,h,a,r]:[t,u,r,e],
               [d,a]:[d,e],
               [d,a,r]:[d,e,r],
               [d,i]:[d,i,a],
               [d,r,a]:[d,r,u],
               [d,u,r]:[d,e,r],
               [d,y]:[d,i],
               [d,y]:[d,i,a],
               [e]:[a],
               [e]:[e],
               [e,b]:[a,b],
               [e,e]:[y,e,a],
               [e,k]:[a,c],
               [e,k]:[e,c],
               [e,k,s]:[e,x],
               [e,l]:[e,a,l],
               [e,l]:[a,l],
               [e,n]:[a,n],
```

[e,s,k,a]:[s,c,a],

[e,t]:[a,t],

[e,v]:[a,v],

[f]:[f],

[f]:[f,f],

[f]:[p,h],

[g,o,o]:[g,u,e],

[g,u]:[g,u,e],

[g,y]:[g,i],

[h,a,r]:[h,e,r],

[h,i]:[h,y],

[i]:[e],

[i]:[i],

[i,a]:[e,a],

[j,a]:[j,u],

[j,a]:[g,e],

[j,a,r,y]:[g,e,r,y],

[k]:[c],

[k]:[n,c],

[k,a]:[c,a],

[k,a]:[c,u],

[k,a,a]:[c,a],

[k,i,a]:[c,i,a],

[k,i,l,o]:[k,e,l,o],

[k,l,a]:[c,h,l,a],

[k,o]:[c,h,o],

[k,o]:[c,o],

[k,r,a]:[c,r,a],

[k,r,i]:[c,r,y],

[k,r,y]:[c,r,y],

[k,s]:[x],

[k,t]:[c,t],

[k,u]:[c,a],

[k,u]:[c,u],

[l]:[l],

[l]:[l,e],

[l]:[l,l],

[l,a,a]:[l,e],

[l,e]:[l,a],

[m,a,k,s]:[m,y,x],

[m,e]:[m,a],

[m,i]:[m,e],

[m,i]:[m,y],

[m,i,y,a]:[m,i,a],

[n,i]:[n,e],

[n,y]:[n,e],

```
            [p,a]:[p,e],
            [p,i,s]:[p,e,s],
            [p,o,k,s]:[p,o,x],
            [r,a,k]:[r,a,c],
            [r,a,k,s]:[r,a,x],
            [r,e,n]:[r,e,i,g,n],
            [r,i]:[r,i,e,e],
            [r,i]:[r,e],
            [s,a]:[c,e,s,s],
            [s,a,r]:[s,u,r],
            [s,h,i,a]:[c,i,a],
            [s,h,o,n]:[t,i,o,n],
            [s,h,o,n]:[s,i,o,n],
            [s,h,o,r,i]:[c,o,r,i,e,e],
            [s,i,s]:[s,c,e,s,s],
            [s,k,s,k,o]:[e,x,c,o],
            [s,t,e,e]:[s,t,i],
            [s,y]:[c,y],
            [t]:[t],
            [t]:[t,e],
            [t,h]:[t,h],
            [t,a]:[t,e],
            [u]:[o,o],
            [u]:[u],
            [v,a]:[v,e],
            [v,e,r]:[v,a,r],
            [v,s]:[v,e],
            [y,d]:[o,i,d],
            [z,i]:[z,e]

        })
    ).

macro(eng_bangla,

    inverse(bangla_eng)

    ).
```

abreshon	keloidalis	keratoakanthoma	bakterial
abcis		keratolisis	boldnes
akanthocis	milia	keratosis	barbay
akny	miliaria	keryon	basal
acrokordon	mohs	lgv	beeard
ektinik	mol	lait	beau's
ekuminata	mol	lamp	bedbagh
ekuminatum	moluskum	lasar	bedsors
aaid	mongolian	legionelosis	benine
elba	mukosil	legionaires	biopsi
ellergic	miksoid	lens	bard
ellergisk	nail	lentigo	barthmark
elopesia	neonatal	leshon	byte
anestheshia	neonatorum	lyce	bytes
engyoma	nurogenic	liken	blefaritis
engular	nurogenic	lifting	blistars
enimal	nevas	linia	blu-gray
enular	neegra	lyns	body
enthraks	neegrikans	leeps	boels
epthos	nipel	local	botoks
ereta	nodosam	loss	botuleenum
erthropod	nodularis	lupes	bruises
ethlet's	nosbleed	lym	boobonik
etopik	notalgia	lymfogranuloma	bag
etipikal	nukai	makul	bumps
evian	noomular	male	barns
besisi	ook	marks	kaf
bebi	objekt	mesels	kalus
hsv	onikolisis	medikashon	kandidiasis
hair	onikomikosis	melanositik	kanker
hairi	onikoshizia	melanoma	kap
hand-fut-and-mouth	oral	melanotik	kapilarytis
hand-fut-and-mauthe	orofatial	melasma	kapitis
hed	ovaruse	mikrobiologikal	karsinoma
hedek	palparbraram	mikrographik	katarakts
heet	papules	mygrane	sel
heetstrok	papulosa	seborik	selulitis
helisis	parasthetika	shingels	sefalik
hemangioma	paronikia	shok	kalazion
harpis	patch	simpleks	kapped
harpetik	patarn	sinas	keilytis
hidradenitis	pediatrik	sixth	chery
hyves	pediculosis	skin	chikenpoks
hordeolum	pedis	smalpox	chyldhud
horn	pemfigas	snekbite	kalamidial
hot	perioral	sok	kolera
heuman	perlish	solar	kondrodarmatytis
hiparpigmentation	fototherapy	solushon	kronic
hyparpigmentation	pilar	sor	kronicus
hiperplashia	pilaris	sors	kivate
hipopigmentation	pink	spidar	klasical
hypopigmentation	pited	splintar	klustar
hipothermia	pitiriasis	spliting	kold
ikthiosis	plaag	spot	koli
ikthiosis	plantar	spots	komon
ilnes	planas	sqames	kondiloma
imunodeficency	plaks	stain	kongenital
impetigo	plaks	estasis	konjunktivytis
infantile	poikilodarma	sting	kontakt
infektion	poison	stings	kontagiosum
	poisoning	siringoma	korn

infektions	polydaktyli	sistemik	korneal
infektiosam	port-wyne	tag	korporis
infektiosam	post-inflamatory	tatu	kosmatic
influenza	pregnansy	telangiktasia	kradel
intartrigo	presbiopia	tenshon	kramps
iritant	preshar	test	kruris
ichh	prikly	thrash	kryosarjery
ivi	primary	tik	kulchar
jelyfeesh	prosidure	tinia	kuretage
jok	pruritik	tung	kutanios
juvenile	sudofolikulitis	toksisity	kutis
keloyd	sudomonas	toksikum	sist
fungal	soriasis	toksin	dandraf
furunkulosis	pubik	trama	dekubitus
jenital	pubis	tritment	dengu
jarman	pastulosis	trench	darmatitis
gonoria	piogenik	tub	darmatofibroma
gaout	rash	tifoid	daramatologik
granuloma	raynaud's	alsar	darmatosis
h5n1	razor	alsars	darmopathy
h1n1	rekarent	unklasifaid	dybetik
hiv	remuval	unkonshasness	dypar
stiches	renual	artikaria	dejit
stroberi	romboidalis	artikarial	diskoid
strech	ringwarm	varisela	desees
stry	rosashia	varikos	drag
stri	rosia	varikositis	drai
stai	rosiola	variola	
swain	rubela	vaskular	erithematosas
sindrom	rubra	veins	erithrasma
sifilis	sakral	venerum	ethnik
dishidrotik	samon	venus	eksaminashon
ekzima	salmonela	versikolar	eksanthem
elektrodesikation	salmonelosis	viral	ekskori
epidarmoid	skabis	viras	eksaushon
erosion	skalp	vitiligo	ai
eruption	skali	valgaris	facial
erithema	skarlet	wart	faciale
strech	skrach	whitlow	faintin
stry	sebashos	wud's	femal
stri	stings	wunds	
stai	stiches	janthelasma	floters
swain	stroberi	jerosis	phlu
sindrom	strech	esst	folikulitis
sifilis	stry	yelow	food
sting	stri	zostar	fudborne
stings	stai	fevar	phut
stings	estasis	fifth	fut
stiches	allargik	farst	fordis
stroberi	hiparhidrosis	flashes	foren
nevas	kensar	flat	frostbyte
akny			

Table 36: An extract of the medical terms in Transliterated Bangla

Roman letter or letter-group	Name	Bangla letter	Unicode
a	AA	আ	\u0986
	SIGN AA	া	\u09BE
b	BA	ব	\u09AC
bh	BHA	ভ	\u09AD
c/ch	CA	চ	\u099A
Ch/chh	CHA	ছ	\u099B
d	DA	দ	\u09A6
dh	DHA	ধ	\u09A7
D	DDA	ড	\u09A1
Dh	DDHA	ঢ	\u09A2
e	E	এ	\u098F
	SIGN E	ে	\u09C7
f	PHA	ফ	\u09AB
g	GA	গ	\u0997
gh	GHA	ঘ	\u0998
h	HA	হ	\u09B9
H	VISARGA	ঃ	\u0983
i	I	ই	\u0987
	SIGN I	ি	\u09BF
I	II	ঈ	\u0988
	SIGN II	ী	\u09C0
j	YA	য	\u09AF
J	JA	জ	\u099C
jh	JHA	ঝ	\u099D
k	KA	ক	\u0995
kh	KHA	খ	\u0996
l	LA	ল	\u09B2
m	MA	ম	\u09AE
M	CANDRABINDU	ঁ	\u0981
n	NA	ন	\u09A8
N	NNA	ণ	\u09A3
Nh	NYA	ঞ	\u099E
ng	ANUSVARA	ং	\u0982
Ng	NGA	ঙ	\u0999
o	A	অ	\u0985
O @ BEGIN	O	ও	\u0993
O @ MIDDLE/END	SIGN O	ো	\u09CB
oi	AI	ঐ	\u0990
	SIGN AI	ৈ	\u09C8
ou	AU	ঔ	\u0994
	SIGN AU	ৌ	\u09CC
oo	SIGN U	ঊ	\u09C1
p	PA	প	\u09AA
ph	PHA	ফ	\u09AB
q	KA	ক	\u0995
r	RA	র	\u09B0
R	RRA	ড়	\u09DC
Rh	DDHA	ঢ়	\u09A2

Roman letter or letter-group	Name	Bangla letter	Unicode
s	SA	স	\u09B8
sh	SHA	শ	\u09B6
S	SSA	ষ	\u09B7
t	TA	ত	\u09A4
th	THA	থ	\u09A5
T	TTA	ট	\u099F
Th	TTHA	ঠ	\u09A0
u	U	উ	\u0989
	SIGN U	ু	\u09C1
U	UU	ঊ	\u098A
	SIGN UU	ূ	\u09C2
v	BHA	ভ	\u09AD
w	UU	ঊ	\u098A
x @ BEGIN	YA	য	\u09AF
x @ MIDDLE/END	KA SA	কস	\u0995 \u09B8
y	YYA	য়	\u09DF
z	YA	য	\u09AF
\	HASANT	্	\u09CD

Table 37: Phonetic Mapping Table [UzZaman 2005]

7 References

Amaral, C., Laurent D., Martins A., Mendes A., Pinto C. 2004. Design and Implementation of a Semantic Search Engine for Portuguese. In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC '04), Lisbon, Portugal. May 2004.

Agarwal, Abhaya; Alon Lavie. 2008. METEOR, M-BLEU and M-TER: Evaluation Metrics for High-Correlation with Human Rankings of Machine Translation Output. In Proceedings of the 3rd Workshop on Statistical Machine Translation (StatMT '08). June 2008. Ohio, USA. Pages 115-118.

Agichtein, Eugene; Steve Lawrence; Luis Gravano. 2001. Learning search engine specific query transformations for question answering. In Proceedings of the 10th International Conference on World Wide Web (WWW '01). Hong Kong, Pages 169-178.

Alam, Firoj; Promila Kanti Nath, Mumit Khan. 2007. Test to Speech for Bangla Language using Festival. In Proceedings of the 1st International Conference on Digital Communications and Computer Applications (DCCA '07), March 2007, Irbid, Jordan.

Arafat, Yeasir; Md. Zahurul Islam, Mumit Khan. 2006. Analysis and Observation from a Bangla News Corpus. In Proceedings of the 9th International Conference on Computer and Information Technology (ICCIT '06), December 2006, Dhaka, Bangladesh.

Bendersky, Micheal; Oren Kurland. 2008. Utilizing Passage-Based Language Models for Document Retrieval. Lecture Notes in Computer Science, Volume 4956, March 2008, SpringerLink, Berlin, Germany.

Beesley, Kenneth R.; Lauri Karttunen. 2003. CSLI Publications 2003. Finite State Morphology.

Bouma, Gosse; Geert Kloosterman, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jorg Tiedemann. 2008. Question answering with Joost at QA@CLEF 2008. In Working Notes for the CLEF 2008 Workshop. Aarhus, Denmark.

Bouma, Gosse; Ismail Fahmi, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jorg Tiedemann. 2006a. The University of Groningen at QA@CLEF 2006: Using syntactic knowledge for QA. In Working Notes for the CLEF 2006 Workshop, Alicante.

Bouma, Gosse; Geert Kloosterman, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jorg Tiedemann. 2007. Question answering with Joost at QA@CLEF 2007. In Carol Peters et al., editor, In Proceedings of CLEF 2007. Springer, Berlin.

Bouma, Gosse; Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jorg Tiedemann. 2006b. Question Answering for Dutch using Dependency Relations. In: C. Peters, F. Gey, J. Gonzalo, H. Mueller, G. Jones, M. Kluck, B. Magnini, M. De Rijke (editors), Accessing Multilingual Information Repositories. Lecture Notes in Computer Science Vol. 4022/2006. Springer. Pages 370-379.

Bouma, Gosse; Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jorg Tiedemann. 2005. Question Answering for Dutch using Dependency Relations. In Proceedings of the CLEF2005 Workshop. 2005.

Bouma, Gosse; Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch, 2001.

Clement, Julien; Philippe Flajolet; Brigitte Valle. 1997. The Analysis of Hybrid Trie Structures. In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '98), November 1997, Philadelphia, USA.

Covington, Micheal A.; 2000. A Fundamental Algorithm for Dependency Parsing. In Proceedings of the 39th Annual ACM Southeast Conference (ACMSE '01), 2001, Georgia, USA.

Dasgupta, Sajib; Abu Wasif; Sharmin Azam. 2004. An optimal way towards machine translation from English to Bengali. In Proceedings of the 7th International Conference on Computer and Information Technology (ICCIT '04), December 2004, Dhaka, Bangladesh.

Day, Min-Yuh; Cheng-Wei Lee; Shih-Hung Wu; Chorng-Shyong Ong; Wen-Lian Hsu. 2005. An Integrated Knowledge-bases and Machine Learning approach for Chinese Question Classification. In Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE '05), November 2005, Wuhan, China.

Doddington, George; 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In Proceedings of the 2nd International Conference on Human Language Technology Research. 2002.

Dong, Z; Q. Dong, 1999 "HowNet", http://www.keenage.com/html/e_index.html/, 1999. Last accessed on November 2009.

Duca, Alan. 2007. The SketchNet Project: Knowledge-based word sense disambiguation. Undergraduate Thesis, May 2007, University of Malta, Malta.

Ferrandez, Sergio; Ferrandez Antonio; Sandra Roger; Pilar Lopez-Moreno; Jesus Peral. 2006.

BRILI, an English-Spanish Cross-Lingual Question Answering System. In Proceedings of the International Multiconference on Computer Science and Information Technology. 2006. pages 25-31.

Finch, Andrew; Eiichiro, Sumita. 2008. Phrase-based Machine Transliteration. In Proceedings of the 3rd International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (IJC-NLP '08), Hyderabad, India. Pages 13-18.

Fleischman, Michael, Eduard Hovy, and Abdessamad Echihabi. 2003. Oine strategies for online question answering: Answering questions before they are asked. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL '03), pages 1-7, Sapporo, Japan.

Harabagiu, Sanda; Steven Maiorano. 1999. Finding answers in large collections of texts: Paragraph Indexing + abductive Inference. In Proceedings of AAAI Fall Symposium on Question Answering Systems, November 1999, Pages 63-71.

Hasnat, Md. Abul; S M Murtoza Habib, Mumit Khan. 2007. A high performance domain specific OCR for Bangla Script. In Proceeding of the International Joint Conferences on Computer, Information and Systems Sciences and Engineering (CISSE '07). December 2007.

Hasan, Fahim Muhammad; Naushad UzZaman, Mumit Khan. 2006. Comparison of different POS Tagging techniques (n-gram, HMM and Brill's tagger) for Bangla. In Proceedings of the International Conference on Systems, Computing Sciences and Software Engineering (SCSS '06). December 2006.

Haque, Nafid. 2006. Web-based English to Bangla Transliteration with lexicon support. Independent Study Project, Spring 2006, BRAC University, Bangladesh.

Haque, Nafid. 2006a. Design of Head-driven Phrase Structure Grammar for Bangla. Undergraduate Thesis, Fall 2006, BRAC University, Bangladesh.

Hossain, Golam Mortuza. 2008. A brief introduction to Anubadok: The Bengali Machine Translator. *Work in progress*. July 2008.

Hsu, Wen-Lian; Shih-Hung Wu; i-Shiou Chen. 2001. Event Identification based on the Information Map. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernatics, October 2001.

Jiang, Long; Ming Zhou, Lee-Feng Chien, Cheng Niu. 2007. Named entity translation with web-mining and Transliteration. In Proceedings of the 20th International Joint Conference on

Artificial Intelligence (IJC-AI '07), Hyderabad, India.

Jijkoun, V B; Hofmann K.; Ahn D D.; Khalid M A; van Rantwijk J.; de Rijke M.; Tjong Kim Sang E F. 2007. The University of Amsterdam's Question Answering System at QA@CLEF2007. Lecture Notes in Computer Science: Advances in Multilingual and Multimodal Information Retrieval. Pages 344-351.

Jurafsky, Daniel; James H. Martin. 1999. Speech and Language Processing, Prentice Hall, 1999.

Karttunen, Lauri. 2000. Applications of Finite-State transducers in Natural Language Processing. In lecture notes in Computer Science. Vol. 2088. Pages 34-46.

Karttunen, Lauri; Kenneth, R. Beesley. 2001. A short history of two-level morphology. Xerox Research Center. September 2001.

Knight, K; Graehl, J. 1997. Machine Transliteration. Computational Linguistics, 24(4), Pages 599-613.

Laurent, Dominique. Patrick Seguela, Sophie Negre. 2005. Cross-Lingual Question Answering using Qristal for CLEF 2005. In Working Notes for the CLEF 2005 Workshop, Vienna, Austria.

Laurent, Dominique. Patrick Seguela, Sophie Negre. 2006. QA better than IR?. In Proceedings of the Workshop on Multilingual Question Answering (MLQA '06), Trento, Italy.

Li, X; D. Roth. 2002. Learning Question Classifiers. In Proceedings of the 19th International Conference on Computational Linguistics (COLING'02), 2002.

Magnini, Bernardo; Matteo, Negri; Roberto, Prevete; Hristo, Tanev. 2001. Multilingual Question/Answering: the DIOGENE System. In Proceedings of the 10th Text Retrieval Conference (TREC '01), 2001.

Mahmud, Altaf; Mumit Khan. 2007. Building a Foundation of HPSG-based treebank on Bangla Language. In Proceedings of the 10h International Conference on Computer and Information Technology (ICCIT '07), December 2007, Dhaka, Bangladesh.

Moldovan, Dan; Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum and Vasile Rus. 2000. The structure and performance of an open-domain question answering system. In Proceedings of the Conference of the Association for Computational Linguistics (ACL '00), 2000.

Monz, Christof. 2003. From Document Retrieval to Question Answering. PhD Thesis. University

of Amsterdam, 2003.

Monz, Christof. 2007. Model tree learning for Query Term Weighting in Question Answering. In Proceedings of the 29th European Conference on Information Retrieval Research (ECIR '07). Lecture Notes in Computer Science 4425, pages 589-596.

Pasca, Marius. 2003. Open-Domain Question Answering from Large Text Collections. CSLI Publications.

Pavel, Dewan Shahriar Hossain; Asif Iqbal, Mumit Khan. 2006. A proposed automated extraction procedure of Bangla text for corpus creation in Unicode. In Proceedings of the International Conference on Computer Processing of Bangla (ICCPB '06). February 2006. Dhaka, Bangladesh.

Papineni, K; S. Roukos, T. Ward, W. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '01), 2001.

Segond, Frederique; Pasi Tapanainen. 1995. Using a finite-state based formalism to identify and generate multiword expressions. In MLTT technical Reports, May 1995.

Strötgen, Robert; Mandl, Thomas; Schneider, René. 2006. A Fast Forward Approach to Cross-lingual Question Answering for English and German. In: Peters, Carol; Gey, Fredric C.; Gonzalo, Julio; Jones, Gareth J.F.; Kluck, Michael; Magnini, Bernardo; Müller, Henning; Rijke, Maarten de (Eds.). Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum, CLEF 2005, Vienna, Austria, Revised Selected Papers. Berlin et al.: Springer [Lecture Notes in Computer Science 4022] Vorab in: Working Notes of the 6th Workshop of the Cross-Language Evaluation Forum, CLEF 2005. Sept. 2005, Wien.

Tapanainen, Pasi. 1995. FSC tool developed at XRCE in 1994-1995.

Terra, Egidio; Charles L.A. Clarke. 2005. Comparing query formulation and Lexical Affinity Replacements in Passage Retrieval. In ELECTRA: Methodologies and Evaluation of Lexical Cohesion Techniques in real world applications, SIGR Workshop, August 2005, Salvador, Brazil.

Tomas, David; Claudio Giuliano. 2009. A semi-supervised approach to question classification. In Proceedings of the 17th European Symposium on Artificial Neural Networks: Advances in Computational Intelligence and Learning, April 2009, Bruges, Belgium.

UzZaman, Naushad. 2005. Phonetic Encoding for Bangla and its application to spelling checker, transliteration, cross-language information retrieval and name searching. Undergraduate Thesis, Spring 2005, BRAC University, Bangladesh.

UzZaman, Naushad; Arnab, Zaheen; Mumit, Khan. 2006. A comprehensive Roman (English) to Bangla transliteration scheme. In Proceedings of International Conference on Computer Processing on Bangla (ICCPB '06), February 2006, Dhaka, Bangladesh.

Van Noord, Gertjan. 2006. At last parsing is now operational. In: Piet Mertens, Cedrick Fairon, Anne Dister, and Patrick Watrin, editors, TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles. Pages 20-42.

Van Noord, Gertjan. 1997. FSA Utilities: A Toolbox to Manipulate Finite-state Automata. In: Darrell Raymond, Derick Wood and Sheng Yu (eds), Automata Implementation. Lecture Notes in Computer Science 1260, Springer Verlag.

Zaanen, Menno van; Luiz Augusto Pizzato; Diego Molla. 2005. Question Classification by Structure Induction. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJC-AI '05), August 2005, Edinburgh, Scotland.

Zhang, Dell; 2004. Web based Question Answering with Aggregation Strategy. In Proceedings of the 6th Asia Pacific Web Conference (APWEB '04), Hangzhou, China. April 2004.

Zhang, Dell; Wee, Sun Lee. 2003. A Web-based Question Answering System. In Proceedings of the SMA Annual Symposium 2003, Singapore, 2003.