Cross-Language Question Answering for Finnish

Lili Aunimo, Juha Makkonen, Reeta Kuuskoski

University of Helsinki, Department of Computer Science,
P.O. Box 26 (Teollisuuskatu 23), FIN-00014 University of Helsinki, Finland
aunimo|jamakkon|rkuuskos@cs.helsinki.fi

Abstract. This paper presents two cross-language question answering systems. They find answers from an English document collection to natural language questions presented in Finnish. These systems are the first open domain question answering systems ever reported to work in Finnish. One of the systems serves as a baseline and it has been designed to be as simple as possible. The other system uses more sophisticated knowledge engineering and human language technologies.

1 Introduction

Question answering (QA) is a task where the information need of the user is formulated as a natural language question and where the answer is given in natural language as well. In general, the length of the answer varies from one word to a couple of sentences, depending on the question. Also those situations where the system is not able to provide an answer should be detected. QA systems can be designed to work on a specific domain only (e.g. an aid for a company help desk [1, 2]), or they can be general purpose systems (e.g. TREC[1] and CLEF QA Tracks [2]). In TREC's QA Track, the questions that are used to evaluate the systems are drawn from a set of real questions posed by end-users. Such sets are the AskJeeves [3] and MSNSearch [4] logs, that have been donated to TREC [3]. In general, question answering systems use unstructured text documents as their database, but, in addition, they can use lists of FAQs (Frequently Asked Questions) and structured databases. In open domain QA systems, the Web is often used as a source of information [4].

Cross-language question answering means that the question is expressed in another language than that in which the documents from which the answer is extracted are written. In this case, the user can use one language to search information from documents written in one or more other languages. This is useful, because it would be tiresome to write the question over and over again in many languages, and also because many users have a good passive knowledge of several languages, but their active knowledge is more restricted [5]. In our

[1] http://trec.nist.gov/data/qa.html
[2] http://clef-qa.itc.it/2004/
[3] http://www.ask.com/
[4] http://www.msnsearch.com/

system, the questions can be expressed in Finnish and the document collection is in English. The system could be extended to handle questions and documents in other languages using the same methodology as presented in this paper. For the simplicity of presentation, we expect from here onwards that the questions and documents are in only one language, and that these languages are different. Cross-language QA is usually implemented either by first applying machine translation to the question and then passing it on to a monolingual QA system or by integrating cross-language processing into the QA system. Our approach is the latter one, because there is no reliable off-the-shelf machine translation software for Finnish. In addition, we expect to improve our results by using the original question as the basis of processing for as long as possible, because when translation is performed, the information content of the question is almost always altered.

The rest of this paper is organized as follows: In Section 2 we present challenges that the processing of Finnish language presents to QA. Next, in Section 3 we describe the overall architecture of our two QA systems. After that each of the main components of QA systems are described in detail. Section 4 descibes the processing of questions, that is, question classification, target concept identification and translation. In Section 5, the information retrieval component of our systems is detailed. Answer processing, which consists of answer extraction pattern instantiation and pattern matching, as well as of answer selection and scoring, is described in Section 6. Section 7 is about evaluation and it presents a comparison of the performance of the two systems. Finally, Section 8 concludes.

2 Question Answering in Finnish

This paper presents the first work on open domain QA for Finnish. Both of the QA systems presented here use Finnish as source language and English as target language. In this section, we describe the special aspects to consider when using Finnish - and not English or any other Indo-European language - as the source language. First, we describe what challenges the morphology of Finnish presents to QA. Then we describe challenges arising from translating Finnish words. Some of the problems are due to the Finnish vocabulary, which has very few common words or word roots with the Germanic and Romanic languages. Another body of problems is caused by word sense ambiguity.

2.1 Morphology

Finnish is a language that differs morphologically from the Indo-European languages. Finnish is mainly considered an agglutinative language, which means that morphemes are glued one after each other to form words [6]. Finnish is a typical language from the Finno-Ugric family in that it contains a large set of grammatical cases and makes extensive use of suffixation. Each Finnish noun, pronoun, adjective and numeral can be inflected in 15 cases. In principle, the number of independent suffixes a word can contain is not limited, but practical

constraints, like ease of pronounciation, of course limit their number. Productive compounding is also a typical feature of Finnish. Finnish is not a purely agglutinative language, which would mean that the morphemes are glued one after each other without any changes. Phenomena like vowel harmony and consonant gradation produce changes determined by context in the morphemes.

For the above mentioned reasons we cannot analyze the questions without first performing morphological analysis. We employ Connexor's [5] functional dependency parser [7] to obtain the mophological analysis. The syntactic parse is needed for question type classification and target concept identification described in section 4.1. In addition, the syntactic chunking is used in *Tikka*'s translation term selection. In the following is an example of the morphological and syntactic analysis: *Minä vuonna se alkoi?* ('In what year did it start?'). It would be parsed as:

#	text	baseform	dependency	morphology
1	Minä	mikä	attr:>2	&A> PRON SG ESS
2	vuonna	vuosi	tmp:>4	&NH N SG ESS
3	se	se	subj:>4	&NH PRON SG NOM
4	alkoi	alkaa	main:>0	&+MV V ACT IND PAST SG3
5	?	?		

From this we see that, for instance, the pronoun *minä* is the essive of *mikä* (what) and that it is an attribute of the nominal head *vuosi* (year).

In QA, we use morphological analysis in the question type classification, question translation and English proper name identification and extraction. Questions are mainly classified into question types using the question words. However, this is not always a straight-forward task. As a simple example, consider the following uses of 'who' (*kuka*):

English	Finnish	question word baseform + case
Who is that man?	*Kuka on tuo mies?*	kuka + NOMINATIVE
Who do you mean?	*Ketä tarkoitat?*	kuka + PARTITIVE
Who is he with?	*Kenen kanssa hän on?*	kuka + GENETIVE
Who do you think he is?	*Keneksi häntä luulet?*	kuka + TRANSLATIVE
Who has it?	*Kenellä se on?*	kuka + ADESSIVE
Who do you trust?	*Kehen luotat?*	kuka + ILLATIVE

In question translation, morphological analysis has to be done before the word can be translated, because dictionaries and translation databases only contain the baseforms of words. The processing of compund words has to be given special attention when automatically translating them from Finnish into English. The compound words may have to be split into several parts before they are given to the dictionary software, because it cannot not contain all the possible compounds. Compounding is very productive in Finnish. In this respect, it can be compared to at least Swedish and German. For example, *the Fire Arms Act* is *tuliaselaki* in Finnish. In the other languages, it was translated using several words in the QA@CLEF 2003 question *When was the National Fire Arms Act*

[5] http://www.connexor.com

approved?: *la Loi sur les armes à feux* (Frence), *la Legge sulle Armi da Fuoco* (Italian) and *el Acta de Armas de Fuego*. In the German and Dutch translations, the English term was used as such [8].

The last task where morphological analysis is used is the identification of proper names that don't need to be translated and the stripping off of the suffixes from them. The discovery of the baseform of proper names is very important when performing automatic translation from Finnish into English. This is especially important, not only in order to be able to form proper answer extraction patterns, but also in order to have the correct query terms in the IR phase. Proper nouns are powerful as search terms [9]. For example, in the question *For which film did Robert Bresson win the Grand Prix at Cannes?* form QA@CLEF 2003, *at Cannes* is *Cannesissa* in Finnish. In the other languages of the multisix corpus [8] containing the QA@TREC 2003 questions it is: *in Cannes* (Dutch and German), *à Cannes* (French), *a Cannes* (Italian) and *en Cannes* (Spanish). Proper names can also coantain other sufffixes besides the grammatical case markers, for example: *Cannesissako* (In Cannes?) *Cannesistammekaan* (Neither from our Cannes).

2.2 Vocabulary

Automatic translation of Finnish words is not a straightforward task, since Finnish and English are not closely related languages. The level of difficulty in the translation part of he system would be a lot lower if we were dealing with closely related languages, like say, Finnish and Estonian or Spanish and Portugese. For example, there is no verb with the meaning *to immigrate* in Finnish, where the concept is expressed as *to come as an immigrant, tulla siirtolaisena*. All the other langauges that participated in the QA@CLEF 2003 had a special verb for *to immigrate*: *emigreren* (Dutch), *einwandern* (German), *immigrer* (French), *immigrare* (Italian) and *emigrar* (Spanish). This word was in the QA@CLEF question *How many Cubans are allowed to immigrate to the United States each year?* [8].

Another difficulty which arises in automatic translation is the semantic ambuguity of words. For example, the word *kerros* (storey) from the QA@CLEF 2003 question *How many storeys high are the twin towers?* has the following 15 translations in our dictionary software: *layer, tier, level, sheet, deck, laminated, flight, coat, floor, storey, story, bed, stratum, film, deep*, of which only three have the same sense as the Finnish word *kerros* in the context of the question.

3 System Architecture

The purpose of a QA system is to provide the user with an answer to the natural language question fed to the system. The answer is searched, for instance, from a text corpus that in our case comprises digital newspaper articles (documents) of 1995 The Glasgow Herald and of 1994 Los Angeles Times. In addition to the

question and document database, a typical monolingual question answering system contains methods for question processing, document retrieval and answer processing [3]. The main goal of question processing is to determine the expected answer type of the question. It is called target concept in this paper. The document retrieval part retrieves documents or passages likely to contain answers to the question using important question words and related terms as the query. The answer processing part performs a match between the question words and the retrieved passages or documents to extract the answer.

We present two QA systems, *Tikka* [6] and *Varis* [7]. *Tikka* is a baseline system that has been designed to be as simple as possible. *Varis* is a more sophisticated system. Figure 1 portrays a general architecture of a question answering system that applies to both *Tikka* and *Varis*. The three ellipses represent the conceptual phases of the question answering process. First the natural language question is translated and turned into a document database query. Then, the relevant documents, i.e., the document in which the answer is most likely to be found, are retrieved from the document database. Finally, the retrieved documents are examined for the answer. In the figure, on the left there are databases and on the right there are modules designed for various tasks relating to these phases. The labeled arrows stand for flow of information.

The phases are covered in more detail in the following sections: In Section 4 we present how we turn the Finnish question into a document retrieval query. Section 5 deals with document retrieval. The third phase, answer extractionis described in Section 6.

4 Question Processing

Before we can look for relevant documents, we have to translate the question to the target language, which in this case is English. In addition to mere translation, we need the type of the question. Thus, we classify each question into one of the predefined classes. Translation and question classification are used both in *Tikka* and *Varis*. In addition to these, *Varis* also performs target concept identification.

4.1 Question Classification and Target Concept Identification

Natural language questions can be assigned classes depending on the type of answer they require. For example, the question *Who is the president of South Korea?* is answered by a person's name, *How many people in U.S. do not have health insurance?* by a number. The Multisix corpus [8] of CLEF-2003 evaluation contains seven question types: *date*, *location*, *measure*, *object*, *organization*, *other*, and *person*. Though there are others, these are the types we focus on in this work. Moreover, the questions are assumed to be correct Finnish.

A natural approach in determining the question type would be to look the question word. It is helpful in many cases, but there are also ambiguous question

[6] 'Woodpecker'
[7] 'Crow'

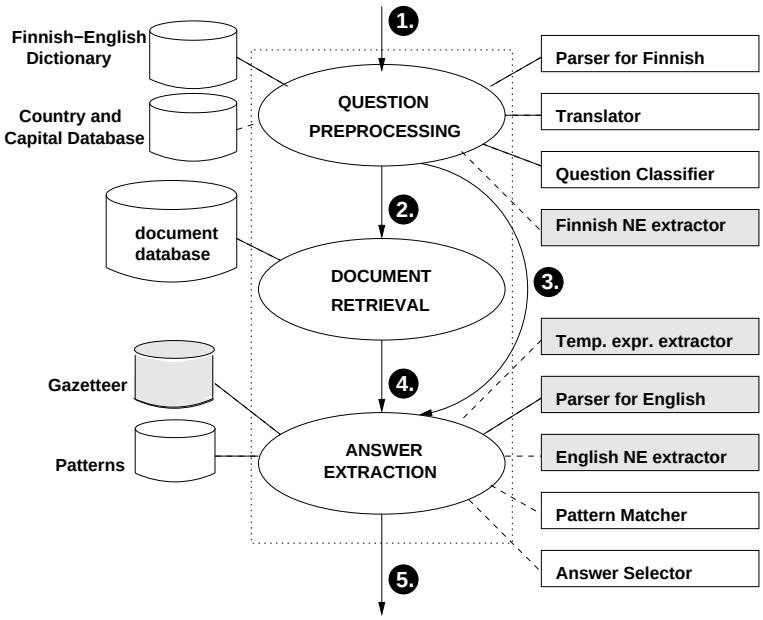

Fig. 1. A general architecture of a QA system. The labeled arrows represent different stages of information flow: 1. the question in Finnish, 2. English translations of the question terms, 3. the question metadata, 4. the set of relevant documents, 5. the answer and the corresponding confidence value. The gray components were used only in *Varis*.

words, and the question classification requires a more thorough examination. Furthermore, not all questions have a question word; rather, they are requests. As we discussed in Section 2.1, Finnish language has characteristics that call for morphological analysis. We employ Connexor's functional dependency parser [7] for Finnish.

In the classification we make use of syntactical information. In some questions, by looking at the complement (*Mikä on Somalian* pääkaupunki*?* [8]) or the subject (*Mitä tarkoittaa* GIA*?* [9]) we may be able derive the question type. A geographical term (river, capital) implies a location, a quantity related noun (size, length, age, value) suggests a measure, a title (mother, president) refers to a person, and an acronym (UNICEF, PLO) often denotes an organization. Sometimes we need to look the attributes of the complement (*Mikä on* Bill Clintonin vaimon *nimi?* [10]) to come up with the type. The noun types are retrieved from a lookup file.

Table 1 lists the classification accuracy (proportion of correct assignments) per question type. What stands out immediately, is the poor results on object-

[8] *What is the* capital *of Somalia?*
[9] *What does* GIA *stand for?*
[10] *What is the name of* Bill Clinton's wife?

type questions. It was difficult to generalize a difference between types object and other for human experts, let alone automatic classification. Therefore, we ignored the type altogether; the questions were assigned to other-type.

type	**correct**	**annotated**	**%**
organization	24	28	0.86
measure	33	34	0.97
date	31	31	1.00
other	18	22	0.82
person	36	39	0.92
object	0	5	0
location	39	41	0.95
total	181	200	0.91

Table 1. The question type classification results.

In addition to the question type, *Varis* extracts the *target concept* of the questions when applicable. By target concept we mean the quality or nature of the topic of the question, which is the focus of interest. In other words, the target concept is the expected answer type of the question. Table 2 contains examples target concepts and question topics. In the first question *How old is Deng Xiaoping?*, the topic is *Deng Xiaoping* and the quality of the topic that is the focus of interest is his *age*.

Question	**Topic**	**Focus of interest**
How old is Deng Xiaoping?	Deng Xiaoping	age
How many islands make up Indonesia?	Indonesia	number of islands
Who is the coach of Indiana?	Indiana	name of coach

Table 2. Examples of question topics and the quality of the topic that is the focus of interest, i.e. the target concept.

The target concept is particularly useful in measure questions. *Varis* utilizes the target concept in the patterns that are discussed in Section 6. Also, *Varis* relies on Connexor's named entity (NE) recognizer in identifying proper names, locations and organizations. In the retrieval phase, proper names are crucial. These pieces of information comprise the metadata denoted by arrow nro 3 in Figure 1.

4.2 Translation

The translation from Finnish to English is based on a dictionary by Kielikone Ltd [11]. The translation proceeds a word by word, so we translate individual words without giving much consideration to the context. However, *Varis* omits words that bear little information with respect to finding the answer or are too common or ambiguous.

Often, when a location, such as a country, is discussed, it is referred to with both a noun (*Somalia, Netherlands*) and an adjective (*Somali, Dutch*). We translate the names of countries and the corresponding adjectives both ways, as a noun and an adjective. The translations reside in Country and Capital Database. The translations of capitals and countries are based on public information provided by Statistics Finland. More elaborate discussion on our approach to translation can be found in [10].

Tikka limits the number of terms translated by translating only nouns and adjective attributes of nouns. In addition, *Tikka* relies on the fact that the entries of the dictionary are mainly ordered by frequency. Thus, it only takes translations from the first sense of the word. Note, that one sense typically has several words. If the first sense is a sense marked as belonging to a special domain, the senses are scanned until one belonging to the common language registry is found.

Since there is very little *a priori* knowledge by which to rank the target language candidates for each question term, *Varis* forms all the possible combinations of the candidates, and then ranks them on the basis of occurrence frequency in the document database. In other words, a Finnish question consisting of n words, each having $t_i, 1 \leq i \leq n$ English translation candidates, adds up to $\prod_{i=1}^{n} t_i$ translation alternatives. Only the translations that do not occur in the corpus are pruned. The questions without proper names or with many commonly occurring words result in exhaustive search. More so, as MG does not enable search with multi-word phrases.

5 Retrieval

We used Managing Gigabytes (MG) [12] [11] for information retrieval task. MG is an open source text indexing and retrieval engine developed as a joint venture of multiple Australian universities. It is capable of indexing large document collections using only a small amount of time and space.

MG, however, has some drawbacks regarding our needs. First, it does not support phrase search or proximity constraints. This made search with compound terms difficult, especially when the terms in the compound were polysemous, e.g. had multiple meanings even though the ortographic form was similar. When these kind of terms were translated from Finnish to English, the amount possible combinations of the terms grew substantially large. The retrieval from the document collection was done in each of these combinations, which made the

[11] http://www.kielikone.fi/en
[12] http://www.mds.rmit.edu.au/mg

document candidate set that had to be processed for answer selection massive. If one could at least limit the possible distance between the query terms in the documents, this phase would not have increased as much.

MG has two query modes, boolean and ranked. In the boolean mode, no special characters (dots, hyphens, dollar signs etc.) can be used, although they might have been of use in some situations. If the query terms could be weighted by their importance, the search could be made more accurate by defining, which of the terms (for instance, proper names occurring in the question) are very important, and which are less important. This is not possible in either of MG's modes, albeit it is not a common feature in any of the standard retrieval engines.

When the target language is English, as it is in both of our QA systems, the document collection consists of texts written in English. If the target language would be Finnish, for instance, MG would not be applicable. This is because of the English alphabet does not recognize scandinavian characters, hence characters ä and ö, which occur in Finnish words are handled as space characters by MG. Therefore the indexing and search using words that contain these letters is not robust.

In search engines, the terms in documents and queries are stemmed in order to be able to match the inflected word forms to the baseform. In stemming, the affices are removed from the word to reveal its stem. In practise, for instance both of the words *developer* and *development* are reduced to the stem *develop*.

MG uses Lovins' stemmer [13], which is developed for stemming English words [12] . The affixes differ between Finnish and English, which causes the truncations performed not to be suitable for Finnish, especially because of the richness of morphology in our language. Since the same kind of stemming is executed both for query and document terms, this is not usually a big problem. When searching for exact matches, however, it might lead to difficulties. For instance, proper noun *Halonen* is reduced to *Halon*, because *-en* as the last morpheme is usually the mark of participe form of a verb and as such an element to be removed in stemming. Finnish inflected forms of the same word can not be combined (the reason for performing stemming at all), either, and therefore documents with words such as *Halosta, Haloselle* are not found with the query term *Halonen*..

6 Answer Extraction

The retrieval phase leaves us with a set of documents. The translated question terms occur in them, but now the task is to find the exact answer. The answer extraction in both systems is based on regular expressions with filled in question-based words. The patterns in *Tikka* and *Varis* are slightly different. *Varis* performs document candidate processing before matching the patterns against the documents, but *Tikka* uses the documents as such.

[13] Available under GPL license at http://sourceforge.net/projects/stemmers/

6.1 Document Candidate Processing

One of the ideas behind *Varis* has been to incorporate external ontologies and natural language processing tools into question answering. Thus, in searching for the answer in the documents we extract proper names and location and temporal information.

The recognition of proper names and locations are based on Connexor's named entity extractor combined with the use of a *gazetteer*, an extensive list of names of people, organizations and locations. The gazetteer is a partly manual compilation of several sources: Statistics Finland [14], Library of Congress [15], CIA Factbook [16] and GEOnet Names Server [17]. The extraction is able to disambiguate the locations using the document as a context. Thus, when encountering the location *Kingston*, the system is able to augment the term with appropriate country, region and continent information.

The extraction and evaluation of temporal expressions is based on our previous work [13]. The temporal expressions are recognized with finite state automata based on dependency functions. The meaning of the expressions is based on a *calendar*: a global timeline and time-units. Usually, the meaning is expressed as an interval on the timeline. Some expressions, *in August 2004* for instance, are explicit in that they require no additional information in order to be evaluated. Then there implicit expressions the meaning of which depends on the utterance time. For this kind of expressions (*two weeks ago*), we use the publication datestamp as the utterance time. The third kind of expressions are evaluated using a reference time: *two weeks later* refers to a prior temporal reference, and the meaning of the expression depends thus on the context.

As a result, the semantical tagging is embedded into the original text, as is shown in Figure 2. The preprocessing is obviously computationally expensive, and therefore we do not preprocess all the documents. We apply the term extraction only on those parts of the text that are likely to contain the answer, i.e., parts in which the question terms occur.

```
After graduating from <loc t='city' l='UK:Europe'>Cambridge</loc> in
mechanical sciences, <ind>Mr Strachan</ind> joined <ind>Alexander
Stephens</ind> of <org>Linthouse</org> as an engineering apprentice
<temp s='19500101' e='19501231'>in 1950</temp>. <temp s='19520101'
e='19521231'>Two years later</temp>, he switched to <org>John Brown
and Co Ltd</org> to complete his apprenticeship.
```

Fig. 2. A simplified example of the semantic tagging during the document preprocessing.

[14] http://www.stat.fi/index_en.html
[15] http://www.loc.gov/
[16] http://www.odci.gov/cia/publications/factbook/
[17] http://earth-info.nga.mil/gns/html/index.html

6.2 Answer Patterns

The answer extraction pattern prototypes of *Tikka* are formed manually by inspecting the QA@CLEF question answer corpus from 2003 [8]. Each question type is given a set of pattern prototypes. The names of classes and the number of pattern prototypes in each class are given in Table 4. In order to obtain pattern instances that can be matched against the documents retrieved, the named slots are filled with the corresponding words from the question translation. As one Finnish word may have several translations in English, each translation combination gets its own pattern instance. The pattern instances are then matched against the documents retrieved and a set of answer candidates is obtained. The answer extraction patterns of *Tikka* are described in more detail in the paper describing the performance of *Tikka* at QA@CLEF 2004 [10]. The selection of one answer from the set of answer candidates is described at a high level in Section 6.3, and in detail in the publication referred to above.

Varis approaches the use of patterns in a slightly different manner. The task of finding the instances of patterns is two-fold: Firstly, the relevant documents are read through with a a three-sentence context window, and each window is rated for relevance using a heuristic distance between the current sentence and the latest occurrences of question terms. If the distance is below a threshold, it means that there are enough question terms present to do employ the document processing described above and then pattern matching. The window spans over multiple sentences, because a number of question terms seldom occur in a single sentence. Secondly, some of the question terms have to occur in the pattern. We call them *anchor terms*, and they often are proper names.

To present the *context heuristic* function more formally: given a context window $C \subset W^n$ of n words and a set of k question words $Q \subset W^k$, we define a function $f_C : W^k \to \mathbb{R}$ such that

$$f_C(Q) = \sum_{w \in Q} \min(d(w), \lambda(w)), \tag{1}$$

where w is a question term, d a distance function $d : W \to \mathbb{N}$ equal to the number of sentences between the current sentence and the latest occurrence of w, and $\lambda : W \to \mathbb{R}$ is a penalty factor of missing w. The penalty is heuristically derived and is based on the type of term: missing the last name of a person or a location is worse than missing an adjective or a verb. If all of the question terms occur in the current sentence, then $f_C(Q) = 0$. Only the contexts rating above a threshold $\theta \in \mathbb{R}$ are passed on to pattern matching.

The patterns themselves are similar to those of *Tikka*. Consider for example the following simple pattern:

```
(\w+) (\w+),* <ind>((\w+| |-)+)</ind>
```

It is supposed to capture occurences of individuals in ways: *Who is the prime minister of UK?* and *Who is John Major?*. In the former case, the name of the individual is asked, and the character strings before the tags have to equal to

prime and *minister*. In addition, the term *UK* has to occur nearby. In the latter case, the title is the target concept, so if the string between the tags matches to *John Major*, the answer is in the first two character strings.

6.3 Answer Selection and Scoring

The patterns are likely to match the text multiple times. How do we know which matching instance is the right one, or at least relevant?

In *Tikka*, the heuristics for selecting the final answer from the possibly quite long list of answer candidates is very simple. The frequency of each answer is calculated, and the one occurring most often is given as an answer. This tactics works well when dealing with answers that occur often in the documents. Examples of this kind of answers are *Boutros Boutros-Ghali* as the head of the United Nations in 1994 and *Kim Young Sam* as the president of South Korea in 1994. If there are several answers with the highest frequency, the first one of them is taken. This works well if the information retrieval engine has been used in ranked mode query mode, because then the documents are sorted by relevance. However, if the information retrieval engine has been used in boolean query mode, then the order is arbitrary. Because boolean queries return a lot smaller set of questions than the ranked one, the effect of this feature is not as bad as it may sound.

After the answer has been chosen, it has to be given a confidence score which tells how sure the system is about the correctness of the answer. *Tikka* only has four confidence scores: 0, 0.25, 0.5 and 1, where 0 means that it is not at all confident and 1 means that it is very confident. A high confidence score is given if the frequency of the answer is very high and/or if the number of other answers is low. The details of the scoring heurisic are presented in the paper describing the performance of *Tikka* at QA@CLEF 2004 [10].

In *Varis*, only the parts of text with occurrences of the question words are examined. The context window spans three sentences, and it may contain several matches of patterns. *Varis* assigns each match a confidence value based the average distance of the question terms from the end of the context window. Multiple instances of the same answer increase the confidence of the answer. The pattern matching runs until there are no documents left or there is an answer supplied with adequate confidence.

7 Evaluation

The evaluation metrics used in evaluating the performance of *Tikka* and *Varis* are mainly the ones that are used in the QA@CLEF 2004 evaluation campaign. They are described in the *Evaluation measure* section of the guidelines[18] of the campain. The main evaluation measure is accuracy, i.e. the proportion of correct answers. The second evaluation measure is the confidence-weighted score.

[18] http://clef-qa.itc.it/2004/guidelines.html

It gives a score between 0 and 1, inclusive, with 1 being a perfect score. The confidence-weighted score rewards systems that can evaluate their own performance. The evaluation metric was introduced at TREC 2002 [14]. In order to obtain a system's confidence weighted score, the answers are sorted according to their confidence score. Then an analog to document retrieval's uninterpolated average pecision is be computed. More formally, if there are N questions in the test set, the confidence-weighted score is:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{number\ correct\ up\ to\ question\ i}{i}$$

In addition to the QA@CLEF 2004 evaluation metrics, we calculated precision and recall for *NIL* questions.

Unofficial results for *Tikka* and *Varis* are shown in figure 3. The results are obtained with the 2003 QA@CLEF data. *Tikka* gave 45 correct answers. One of the answers was inexact and there were no unsupported answers in the answer set.

	Tikka		Varis	
	Absolute	**Percentage**	**Absolute**	**Percentage**
Accuracy	45/200	22.5	58/200	29.0
Confidence-weighted score	0.38		0.44	
Number of NIL answers	132/200	66.0	75/200	37.5
Precision of NIL answers	13/132	9.8	12/75	16.0
Recall of NIL answers	13/15	86.7	12/15	80.0

Table 3. The unofficial results for *Tikka* and *Varis* with QA@TREC 2003 data.

Varis gave 58 correct and 13 inexact answers. The evaluation was strict: For instance, when asked the number of Puerto Ricans in Los Angeles, the answer 40,000 was considered inexact (correct: *more than 40,000*). When asked the home town of Edwin Tang, *New York* was considered inexact (correct: *New York City*). When asked the location of Biosphere 2, *Tucson* was inexact, since the correct answer was *outside Tucson*. In two cases, the answer lacked the state, though the city was correct.

In the confidence-weighted score, the order in which the equal confidence scores are sorted has significance, because the denominator iterates from 1 to N. For example, for answers with confidence 1.0, having the correct answers before the incorrect ones results in higher score than in reverse order. We sorted the judgements having the same confidence by the question id.

Table 4 shows the correct answers per question type as well as the number patterns in use. *Varis* has more patterns than *Tikka*, which is partly due to the embedded semantic annotation; they make the patterns more 'rigid'.

We would have expected higher accuracy from *Varis* with the date-type questions. The semantic annotation of temporal expressions did not benefit as much

Table 4. Question type classes, number of prototype patterns in each class, number of questions belonging to each class in QA@CLEF 2003, number of correct answers in each class and percentage of correct answers in each class.

		Tikka			*Varis*		
type	**questions**	**patterns**	**correct**	**%**	**patterns**	**correct**	**%**
date	31	3	6	19.4	15	10	32.2
location	41	18	19	46.3	32	13	31.7
measure	34	22	11	32.4	22	13	38.2
person	39	16	7	17.9	11	9	23.1
object	5	0	0	0.0	0	0	0.0
organization	28	0	1	3.6	16	7	25.0
other	22	5	1	4.5	19	6	27.2
total	200	64	45	22.5	115	58	29.0

as was initially thought. Maybe the augmented information increased the risk of false-alarms. The correct answer did not rank high enough, as there were more instances matching patterns. Similar disappointment results from location-questions: *Tikka* having fewer patterns and no semantic annotation exceeds *Varis* in performance. For some of the cases, as mentioned with the inexact answers, we have the incomplete patterns to blame; states and additional attributes are omitted. Clearly, *Tikka*'s frequency based answer selection works better with location-questions than that of *Varis* that is plagued by the wieldy document processing.

8 Conclusions and Future Work

To the best of our knowledge, the work presented in this paper is the first time cross-language QA has been done using Finnish as a source language. Altogether, there has been very little work on any type of QA for Finnish. Keeping this in mind, it was interesting to get the baseline system up and running and to observe that it could answer 22.5 % of the questions presented to it correctly. Our other system, which required more sophisticated knowledge engineering and human language technologies, attained an accuracy of 29.0 %.

Due to the very different nature of Finnish in comparison to the Indo-European languages, special attention has been paid to the morphological analysis of the question words and to question translation.

For both of our QA systems, the high number of *NIL* answers suggests that there should be more patterns in the pattern database. Analyzes of the systems show, that documents containing the answers are found, but there is not enough patterns to match the results. In further work, it is crucial to develop methods that enable more efficient and easy generation of answer extraction patterns. This is essential also when QA systems are ported to other languages and domains.

Besides too few prototype patterns, another main source for errors in the baseline system is the translation phase. Its heuristic for choosing the correct

translations among the many candidates is far too simplistic. Better word sense disambiguation technology is needed to improve accuracy.

References

1. Aunimo, L., Heinonen, O., Kuuskoski, R., Makkonen, J., Petit, R., Virtanen., O.: Question answering system for incomplete and noisy data: Methods and measures for its evaluation. In: Proceedings of the 25th European Conference on Information Retrieval Research (ECIR 2003), Pisa, Italy (2003) 193 – 206
2. Busemann, S., Schmeier, S., Arens, R.G.: Message classification in the call center. In: Proceedings of 6th Applied Natural Language Processing Conference, Seattle, Washington, USA (2000)
3. Voorhees, E.M.: Overwiew of the TREC-2003 Question Answering Track. In Voorhees, E.M., ed.: Proceedings of TREC-2003, Gaithersburg, Maryland, Department of Commerce, National Institute of Standards and Technology (2003)
4. Fleischman, M., Hovy, E., Echihabi, A.: Offline strategies for online question answering: Answering questions before they are asked. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan (2003)
5. Gonzalo, J.: Scenarios for interactive cross-language retrieval systems. In: Proceedings of the Workshop 1: Cross-Language Informaion Retrieval: A Research Roadmap Workshop held at the 25th Annual International ACM SIGIR Conference, Tampere, Finland (2002)
6. Comrie, B.: Languages Universals and Linguistic Typology. Blackwell Publishers (1989)
7. Järvinen, T., Tapanainen, P.: A dependency parser for english. Technical Report TR–1, Department of General Linguistics, University of Helsinki (1997)
8. Magnini, B., Romagnoli, S., Vallin, A., Herrera, J., Penas, A., Peinado, V., Verdejo, F., de Rijke, M.: The Multiple Language Question Answering Track at CLEF 2003. In Peters, C., ed.: Working Notes for the CLEF 2003 Workshop, Trondheim, Norway (2003)
9. Pirkola, A., Järvelin, K.: Employing the resolution power of search keys. Journal of the American Society for Information Science and Technology **52** (2001) 575–583
10. Aunimo, L., Kuuskoski, R., Makkonen, J.: Cross-language Question Answering at the University of Helsinki. In Peters, C., ed.: Working Notes for the CLEF 2004 Workshop, Bath, United Kindom (2004) To appear.
11. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images. second edn. Morgan Kaufmann Publishers (1999)
12. Lovins, J.B.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics **11** (1968) 22–31
13. Makkonen, J., Ahonen-Myka, H.: Utilizing temporal information in topic detection and tracking. In Koch, T., Solvberg, I.T., eds.: Proceedings of 7th European Conference on Digital Libraries (ECDL 2003), Trondheim, Norway, Springer-Verlag (2003) 393–404
14. Voorhees, E.M.: Overwiew of the TREC-2002 Question Answering Track. In Voorhees, E.M., Buckland, L.P., eds.: Proceedings of TREC-2002, Gaithersburg, Maryland, Department of Commerce, National Institute of Standards and Technology (2002)