



UPPSALA  
UNIVERSITET

Department of Linguistics and Philology  
Språkteknologiprogrammet  
(Language Technology Programme)  
Master's thesis in Computational Linguistics

# Question categorization for a question answering system using a vector space model

Anna Hedström

Supervisor:  
Tomas Englund, Dialogue Technologies AB  
Mats Dahllöf, Uppsala University

## **Abstract**

The purpose of the thesis is to automatically assign questions into sense categories, where the sense categories are represented by predefined paraphrase sets of questions. The paraphrase sets are available for each domain in a Question Answering system. A Java program was developed for the categorization of the questions. The program implements a vector space model for representing and comparing the terms in the questions.

The Java program supports a method for extracting different features for indexing the vectors. The features used as index terms are keywords expanded with synonyms which are retrieved from the domain lexicon for the Question Answering system, unigrams, and bigrams of words or characters. Experiments with removing stop words from the questions in order to reduce the number of index terms were also performed.

The program has been evaluated for two different domains for the Question Answering system. The highest accuracy rate accomplished for one domain was 62% and 29% for the second domain. The difference in the results is mainly due to the high number of paraphrase sets used during the categorization for the second domain. In experiments where the numbers of paraphrase sets were reduced, the accuracy rate improved to 40%.

The task was proposed by Dialogue Technologies AB, a company which provides Question Answering systems. All the user questions to the Question Answering applications are saved in log files. To be able to retrieve information about what questions the users ask, automatic categorization of the log files is essential.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vi</b>
<b>Acknowledgments</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Purpose of the thesis . . . . .	2
1.2 Outline . . . . .	2
<b>2 Information retrieval and text categorization</b>	<b>3</b>
2.1 Indexing . . . . .	4
2.1.1 Selection of features . . . . .	4
2.1.2 Term weights . . . . .	5
2.2 N-gram based categorization . . . . .	6
2.3 The vector space model . . . . .	7
2.3.1 Similarity functions . . . . .	8
<b>3 The Phoenix system</b>	<b>10</b>
3.1 System overview . . . . .	10
3.1.1 Schemes . . . . .	10
3.1.2 Entity and term . . . . .	11
3.1.3 Lexicon . . . . .	11
3.1.4 The Phoenix parser . . . . .	11
<b>4 Experiments</b>	<b>13</b>
4.1 Data . . . . .	14
4.1.1 Predefined paraphrase sets of questions . . . . .	14
4.1.2 Pre-processing of the log files . . . . .	15
4.2 Indexing methods . . . . .	16
4.2.1 Indexing with keywords and synonyms . . . . .	16
4.2.2 Indexing with $n$ -gram . . . . .	18
4.3 The vector space model program . . . . .	19
4.3.1 Lexical analysis . . . . .	20
4.3.2 Creation of a template vector . . . . .	20
4.3.3 Creation of vector instances . . . . .	21

4.3.4	Tf.idf weighting of the terms . . . . .	21
4.3.5	Cosine comparison and categorization . . . . .	21
4.3.6	Final grouping of questions of the log files . . . . .	22
<b>5</b>	<b>Evaluation</b>	<b>23</b>
5.1	Manual categorization of a key . . . . .	23
5.2	Evaluation measures . . . . .	24
5.3	Results . . . . .	25
<b>6</b>	<b>Conclusions</b>	<b>29</b>
	<b>Bibliography</b>	<b>32</b>
<b>A</b>	<b>Evaluation of paraphrase sets, Glocalnet.</b>	<b>33</b>
<b>B</b>	<b>Distribution of numbers of questions to each paraphrase sets for the Glocalnet.</b>	<b>34</b>
<b>C</b>	<b>Distribution of numbers of questions to each paraphrase sets for the SOIC domain.</b>	<b>35</b>
<b>D</b>	<b>Sample from categorized log file, questions assigned to paraphrase set 0.</b>	<b>36</b>

## List of Figures

2.1	A weighted term by document matrix. . . . .	8
2.2	A two dimensional vector space. . . . .	9
4.1	Project outline. . . . .	13
4.2	Steps in the vector space model program. . . . .	19

## List of Tables

4.1	Sample from the log file for the SOIC domain. . . . .	17
4.2	Sample from the log file for the Glocalnet domain. . . . .	18
5.1	Features used for indexing. . . . .	23
5.2	Question distribution over paraphrase sets in the key for the Glocalnet domain, total number of paraphrase sets is 158, total number of questions is 150. . . . .	25
5.3	Question distribution over paraphrase sets in the key for the SOIC domain, total number of paraphrase sets is 244, total number of questions is 200. . . . .	25
5.4	Results for the Glocalnet domain, question-to-collection comparison. . .	27
5.5	Results for the Glocalnet domain, question-to-question comparison. . .	27
5.6	Results for the SOIC domain, question-to-collection comparison. . . . .	27
5.7	Results for the SOIC domain, question-to-question comparison. . . . .	27
A.1	Glocalnet: Distribution of number of question to the paraphrase sets with at least 3 % of the question assigned to them. The table shows precision and recall for index method using <i>unigrams with stop words</i> and <i>bigrams of characters with stop words</i> as features. . . . .	33

# Acknowledgments

I would like to express my gratitude to some people for their help while I wrote this Master's thesis:

Thanks to everybody at Dialogue Technologies AB, especially Tomas Englund, for helping me a lot during the writing process, giving very insightful comments regarding the text.

Also, I would like to thank my supervisor at the Department of Linguistics and Philology, Mats Dahllöf, for suggesting a number of improvement to the text and for answering questions of all sorts.

Thanks to Per Johnsson for help and support with Java and Prolog programming. Also, thank you Per for always taking the time to discuss whatever problems arose or just for listening and encouraging me. For your immense patience and invaluable support every step of the way, I am eternally grateful.

My friends at the Language Technology Program, especially Eva Ericsson, Maria Moutran Assaf and Sumithra Velupillai. Thanks for being my dearest friends and for great support during the accomplishment of the thesis. Thanks to Sumithra for reading trough different versions of the thesis, giving helpful comments.

# 1 Introduction

Text categorization is a fundamental part of Information Retrieval and Natural Language Processing. In text categorization, a new text document is assigned to one of a pre-existing set of document classes. The term document categorization includes different techniques for automatic grouping of text, for example clustering, a statistical method used to find natural groupings in text (Sebastiani, 1999). Text categorization is for example used in language detection, text-genre categorization and subject identification. It is also used in information retrieval to improve information retrieval systems and browse collections of texts.

When dealing with large text collections a manual classification is not feasible since it is a very time-consuming task. Furthermore, it is a well-known fact that human classifiers in some cases makes different decisions about what is the proper category for a document. Categorization of documents is essential when dealing with large document collection and sorting the document collections makes the overview of the corpora easy and manageable.

The first experiments on solving the categorization problem in an automatically way was done by using rule based techniques. Today, the standard approach is to use supervised machine learning where classification algorithms are trained. The document is often represented in a word-based space (Jurafsky and Martin, 2000; Cavnar and Trenkle, 1994; Allan and Bekkerman, 2003).

The purpose of this thesis is to find a way to automatically assign large collections of natural language questions into paraphrase question sets. The task was proposed by Dialogue Technologies AB, a company which develops Question Answering systems. In the Question Answering system that Dialogue Technologies AB supplies, a user can ask a question in natural language and an answer is retrieved from a relational database. The number of questions to the applications supplied by Dialogue Technologies AB is about five million questions per year and per application. Every question to the Question Answering system is stored in a log file. To measure how well the system works a recall rate is calculated, i.e. how many of all incoming questions the system is able to answer. Even if the system has a high recall rate, about 90 percent, it will still generate 500 000 non-answered question per year, per application. When the Question Answering system is upgraded, the questions in the log files over non-answered questions contain important information about the system. One example of such information is what type of question the user asks. To facilitate the upgrading of the domains, it is important to find out if there are many questions about new areas, which were not predicted from the beginning, which the system cannot answer yet. In order to find patterns in the large amount of questions, automatic categorization is necessary.

## 1.1 Purpose of the thesis

The purpose of this thesis is to automatically assign questions to sense categories, where the categories are represented by predefined paraphrase sets of questions. The paraphrase sets are available for each domain in the Question Answering system and consist of different framings of questions that will lead to the same answer in the Question Answering system.

The categorization will be done by using a vector space model. The purpose of the thesis is also to perform experiments with different features for indexing the vectors. The features are keywords expanded with synonyms, retrieved from the domain lexicons in the Question Answering system, unigrams and bigrams of words or characters. The material consists of logfiles with non-answered questions from a Question Answering system. The result will be evaluated by measuring accuracy, precision and recall.

## 1.2 Outline

The thesis is divided into four main parts. The first part introduces the subject and gives a description of the purpose of the thesis. In the second part, chapters two and three, a theoretical background to text categorization and the vector space model are given. A brief overview of the Phoenix system is also given in chapter three. The third part, chapter four, describes the experiments. This includes pre-processing of log files and the paraphrase sets of questions. The indexing process and the Java program made for the categorization process are also described in chapter four. In the final part, chapter five and six, the results are presented and analyzed and future work is discussed.

## 2 Information retrieval and text categorization

The common approach to text categorization is to use supervised machine learning, which means that the categorization is based on documents that already are assigned to a category. By learning characteristics for the categories, a text categorizer is built. (Sebastiani, 1999). When the categorization process assigns exactly one category to each document in the collection it is called single-label categorization. In single-labeled categorization, the categories that the documents are divided into never overlap. In contrast, there is multi-label categorization where a number of category labels (none or several) can be assigned to a document (Sebastiani, 1999).

Text categorization is also used in information retrieval. A similarity between text categorization systems and information retrieval systems is that both systems often use word-based spaces to represent documents and queries, most often a vector space model. A vector space model represents text in a formal manner by the use of multi-dimensional space. Each dimension in the multi-dimensional space corresponds to the words in the document collection. The vectors are the representation of the texts, based on term occurrences and the vector denotes a point in that space (see 2.3). Consequently, both text categorization and information retrieval systems in some cases rely on the comparison between the document vectors and a query vector. In a document retrieval system the user asks a query and the query can be seen as the translation of an information need, by which the user is led to a document. In text categorization, the equivalent of the query is the result of the analysis of the texts that belong to the target class and texts that do not. According to Krüger-Thielmann et al. (Krüger-Thielmann and Pajmans, 2005) both information retrieval and text categorization can be seen as to consist of four phases:

- Indexing: Each document is described by a set of representing keywords, called the index form. The content of the document is translated into symbols and structures of the index language. This is done by creating a document representation of every document. The representation consists of selected features such as keywords or key phrases.
- Query formulation or categorizer formulation: The information need, for example a user query, is also translated into the index language. This means that the same representation is used for the document and the query or categorizer.
- Comparison: The query (or the categorizer) is compared with the document representation and a binary or a graded similarity is computed. Categorization systems will most often require a binary decision which shows if the document to be categorized belongs to the category or not.

- Feedback: The system is evaluated in some way. Common measures for evaluation of text categorization systems and information retrieval system are accuracy, precision and recall.

Text classifiers can be divided into two types: binary categorizers and  $m$ -ary categorizers ( $m > 2$ ). A binary categorizer makes a YES/NO decision for each category, given a document. An  $m$ -ary categorizer produces a ranked list of candidate categories for each document with a confidence score for each candidate. Often when a confidence score is used, a threshold is defined for the system. The threshold states the confidence score has to be at a specific score to be accepted as a correct categorization (Yang, 1999).

The following terminology will be applied throughout the thesis (Jurafsky and Martin, 2000; Manning and Schütze, 2002):

- A term refers to a lexical item that occurs in a collection which also can include phrases.
- A collection refers to a set of documents being used to satisfy user requests.
- A document usually refers to the unit of text indexed in the system which is available for retrieval.
- A query represents a users' specific need expressed as a set of terms.

## 2.1 Indexing

The indexing process usually consists of the steps term extraction and term weighting. The selection of which features, i.e. keywords or key phrases, to use in the indexing language is called term extraction. Term weighting is used to decide how much impact the term will have during the rest of the categorization. The weight is decided by a function such as term frequency. These two steps will be described in the following sections.

### 2.1.1 Selection of features

In many text categorization systems the meaning of documents is captured based only on the words within the document, not the ordering and the syntactic functions of the words. In such systems "*I see what I eat*" and "*I eat what I see*" would mean exactly the same thing (Jurafsky and Martin, 2000, p.647). An approach where the syntactic information is ignored is called a bag-of-words method. The vector space model is an example of a bag-of-words method (Jurafsky and Martin, 2000).

The index term is a word whose semantics represents the documents main theme. Index terms are often nouns but can also be other things than word types such as  $n$ -grams or phrases (Yatez et al., 1999). If the system uses word types as features the system can store all words in the same form as they occur in the text, but, most often, the features need to be adapted. To adapt the features, different techniques can be applied during the term extraction phase (Krüger-Thielmann and Pajmans, 2005).

The most common technique for adapting features is to reduce the number of index terms. Using all words in a collection to index the document will generate too much noise for the retrieval task (Yatez et al., 1999). It is possible to filter out function

words for example by using a list of stop words. The motivation for this is that high frequency, closed-class words are seen as carrying little semantic weight, and rarely contribute information to help in a word-by-word match or when deciding similarities between sentences. Because of this, those words are not likely to be considered as keywords (Jurafsky and Martin, 2000).

The system can also bring down the number of items to be indexed by applying a truncation or stemming algorithm or by using lemmatization (Hulth, 2003; Kyoung-Soo and Hoojung, 2004-13). Stemming and lemmatization causes a mapping of several morphologically related words to the same index entry. The fundamental question of stemming and lemmatization is if variants of lexical items should be collapsed into a single root form with a single summarized frequency count, or treated as distinct items with separate term frequencies. While treated as distinct items, the number of index terms will be larger than if the former is used (Jurafsky and Martin, 2000).

On the other hand, when a document is retrieved for a question a problem is that queries usually only consists of a small number of content words and documents are longer texts. To perform a linguistic analysis, more textual material is normally required to model the query and the results for mechanisms based on statistical analysis are higher given more input. A solution to that problem might be to add synonyms to the terms. Lexical resources such as WordNet or a thesaurus are often used for this purpose. In such resources the semantic relations between words can be found (Karlgrén, 2000).

There are two types of indexing methods. The first is derived indexing where the features are taken from the document and they can be words, n-grams, collocations or any other feature that is considered as useful for the division of documents in the set of relevant and non-relevant. Another type of indexing is assigned indexing, where the keywords are taken from an independent list of terms, classification system or ontology. The keywords can for example be expanded with synonyms from WordNet (Krüger-Thielmann and Pajmans, 2005).

### 2.1.2 Term weights

After the term extraction has been made, the terms are weighted. The weight of the term decides how much impact the term will have during the rest of the categorization process. For the vector space model, frequency-based weights are used.

Not all terms are equally useful for describing the content of a document. A word which appears in every document in a collection is of no use for the categorization task, because it tells us nothing about the specific content in a document. Words that appear in few documents are better to use as index terms. Terms which occur frequently within a document may reflect its meaning more strongly than terms that occur less frequently and should thus have higher weights (Krüger-Thielmann and Pajmans, 2005; Yatez et al., 1999).

The final weight that is stored for an individual word-document combination may be computed in two ways, either as plain word weights or word-document weights. The plain word weight is only related to the keyword itself. This means that it will be the same for all occurrences of a keyword and the computed word weight will remain constant in the database. For the word-document weighting, the properties of the individual document are also taken into consideration. When using word-document weighting, a document word weight that will be different for every new document

is computed. The vector space model can be used with both plain word weights and word-document weights.

Two examples of plain word weights are the Poisson model and the Discrimination value. Plain word weights are not used in this thesis and will therefore not be discussed further (Krüger-Thielmann and Pajmans, 2005).

There are three frequency variables that are combined in forming the final word-document weight for a keyword; term frequency is defined as the number of keywords  $i$  in the document  $j$ , document frequency is defined as the number of documents in which the keyword  $i$  occurs and collection frequency is defined as the total number of times that the keyword  $i$  occurs in the collection.

$$tf_{ij} = \text{term frequency of term } i \text{ in document } j \quad (1)$$

$$df_i = \text{document frequency of term } i \quad (2)$$

$$cf_i = \text{collection frequency of term } i \quad (3)$$

The  $tf$  measure can be a measure of how salient a word is within a given document. The higher value of the  $tf$ , the more likely it is a good descriptor of the content in the document (Manning and Schütze, 2002). When the weights in the vector are word frequencies, the longer the document, the bigger the chance is that this keyword occurs many times. Therefore a normalization or a correction of the plain term frequency might be needed. One way to accomplish this is to divide the term frequency by length of the document. The length of the document can be measured in several ways. One example is to count the characters without applying filters of any kind (Krüger-Thielmann and Pajmans, 2005). The term can also be weighted after the square root of the frequency in the document,  $f(tf) = \sqrt{tf}$ .

When a semantically focused word occurs in a document, it often occurs several times. A word that is not semantically focused has a distribution that is spread over all documents (Manning and Schütze, 2002). What therefore is needed is a measure that favors terms which occur in few documents. The fewer documents a term occurs in, the higher the weight. This can be done by combining the document frequency with the term frequency. One example, and the most popular of a word-document weighting measures is the  $tf.idf$  weighting schema, which combines the term frequency ( $tf$ ) and the inversed document frequency  $1/df$  to one weight.

$$idf_i = \text{inverse document frequency for term } i \text{ in document } j \quad (4)$$

(Krüger-Thielmann and Pajmans, 2005).

The final weight for a term  $i$  in document  $j$  in the document collection  $D$  is the product of the two weights:

$$weight_{ij} = tf_{ij} \cdot idf_i \quad (5)$$

(Jurafsky and Martin, 2000)

## 2.2 N-gram based categorization

Experiments have shown that word-based representations, such as bag-of-words are very effective in document representations. The main drawback of the bag-of-words

representation is in the destruction of semantic relations between words. Since collocations are also lost in the bag-of-words representation, the meaning of the collocations are lost. To overcome this drawback, using  $n$ -grams is a way to enrich the bag-of-words approach to the document representation for example by incorporating bigrams in the document representation, so called bag-of-bigrams.

$N$ -grams can be constructed from characters or words. Methods based on word  $n$ -grams are more stable than character-based methods because individual characters or phonemes are often indistinct and co-articulation is facilitated when matching longer segments (El-Nasan et al., 2001).

There is always a set of words which dominates most of the other words of the language in terms of frequency of use. The same holds for  $n$ -grams. When doing categorization with frequency statistics and comparing documents from the same category, the documents will have similar  $n$ -gram frequency distribution. One improvement to the  $n$ -gram approach is to omit the statistics for  $n$ -grams which are very common. This would give a better discrimination from those statistics of the  $n$ -grams that remain (Cavnar and Trenkle, 1994).

Another disadvantage for using bigram based categorization might be that most of the bigrams are no more informative than just random combinations of unigrams. Highly discriminative bigrams could improve the categorization results but their contribution in comparison to what a high number of unigrams can contribute, is low. Highly discriminative bigrams do exist but their ratio to “junk” bigrams might be low. Allan et al argues that in domains with severely limited lexicons and high chances of constructing stable phrases the bag-of-bigram method can be useful (Allan and Bekkerman, 2003).

There are experiments that show that information retrieval systems with index features based on  $n$ -grams has an advantage in retrieval over index features based on unigrams the longer the question is (Mayfield and McNamee, 1997).

## 2.3 The vector space model

The vector space model is widely used for retrieval purposes. It uses an underlying metaphor of spatial proximity to decide semantic proximity. The vector space model proposes a framework where partial matching between documents or documents and queries is possible. The weights of the terms in the vectors are used to compute the degree of similarity between the documents. The correlation between the vectors can be quantified by measuring the cosine of the angle between two vectors (Yatez et al., 1999; Manning and Schutze, 2002).

During the indexing process, the document content is transferred to the vector representation. Documents and queries in the vector space model are represented as vectors of features representing the terms that occur within them, i.e. within the collection. Each document is represented by an  $n$ -dimensional vector, where  $n$  is the number of index terms in the collection. The value of each feature indicates the presence or absence of a given term in a given document. Hence, it is possible to determine the relevance of a document to a query (or document to document) by determining the number of terms they have in common. The elements in the vectors may also have any weight attached to it (Jurafsky and Martin, 2000).

The database can be represented as a term-document matrix with documents as rows of  $n$  distinct terms,  $t_1, t_2, \dots, t_n$ , each column represents the assignment of a spe-

$$\begin{bmatrix} w_{11} & w_{21} & \dots & w_{i1} & \dots & w_{n1} \\ w_{12} & w_{22} & \dots & w_{i2} & \dots & w_{n2} \\ \vdots & \vdots & \dots & \vdots & \dots & \vdots \\ w_{1j} & w_{2j} & \dots & w_{ij} & \dots & w_{nj} \\ \vdots & \vdots & \dots & \vdots & \dots & \vdots \\ w_{1m} & w_{2m} & \dots & w_{im} & \dots & w_{nm} \end{bmatrix}$$

**Figure 2.1:** A weighted term by document matrix.

cific term to the  $m$  documents  $d_1, d_2, \dots, d_m$  of the collection. For every document-keyword combination there can be a weight, where the weight may be zero or one for binary vectors or a real for weighted vectors,  $w_{ij}$ .

In the vector space model queries are resolved by translating the query to a vector with the same range as the documents, comparing them and ranking the similarities (Manning and Schütze, 2002).

### 2.3.1 Similarity functions

In the vector space model documents are ranked by computing the similarity with a query. The vector similarity for both binary and real-valued vectors can be measured by the cosine measure. Figure 2.2 shows an example of a vector space with two dimensions. The two dimensions corresponds to the terms “carrot” and “candy”. One question and three documents are represented in the space. Document 2 has the smallest angle with the question “carrots are candy”. This is because both the terms “carrot” and “candy” are salient in that document and therefore have a high weight. In the other two documents, the terms occur but are not as important.

The cosine formula calculates the cosine of the angle between two vectors and is the most common measure for comparing vectors. The cosine ranges from 1.0 which corresponds to  $\cos 0^\circ$  for vectors pointing in the same direction and over 0.0 for orthogonal vectors, which corresponds to  $\cos 90^\circ$ . -1.0 corresponds to  $\cos 180^\circ$  for vectors pointing in the opposite direction (Manning and Schütze, 2002).

If the vectors are normalized<sup>1</sup>, the cosine can be computed by using the dot-product:

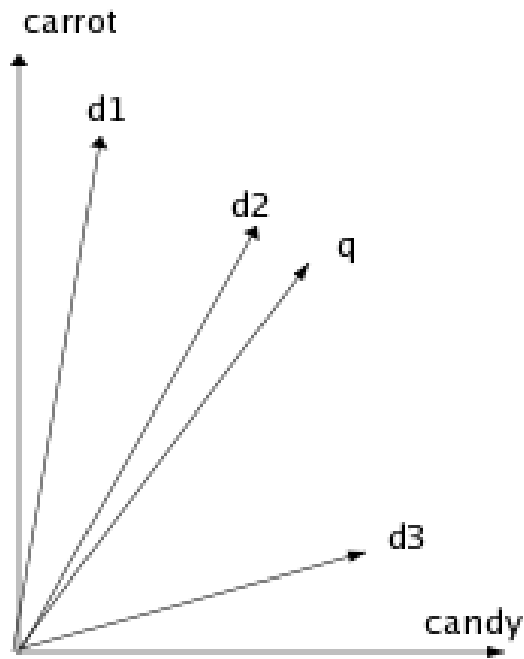
$$\text{dot}(d_j, d_k) = \vec{d}_j \bullet \vec{d}_k = \sum_{i=1}^N w_{i,k} \times w_{i,j} \quad (7)$$

Normalizing the vectors eliminates the importance of the exact length of a document’s vector in space. A longer vector, corresponding to a longer document, would have an advantage over shorter ones and get a higher rank if normalization is not

<sup>1</sup>A vector is called normalized if it has a unit length according to the Euclidean norm. The Euclidean length of a vector is defined as:

$$\vec{x} = \sqrt{\sum_{i=1}^N x_i^2} \quad (6)$$

where  $n$  is a sequence of real numbers, the number of dimensions in the vector and  $x_i$  denotes the  $i^{\text{th}}$  component of  $\vec{x}$  (Manning and Schütze, 2002)



**Figure 2.2:** A two dimensional vector space.

performed. The normalization process can be performed by converting all vectors to a standard length by dividing each of their dimensions with the overall length of the vector:

$$\sqrt{\sum_{i=1}^N W^2_{i,k}} \tag{8}$$

The cosine computes how well the occurrence of terms correlates in a query and a document (or document to document). When using the dot product, terms are assumed to be uncorrelated where the term vectors are orthogonal and their vector dot product is equal to 0 (Jurafsky and Martin, 2000) (Manning and Schutze, 2002).

## 3 The Phoenix system

The material used for the categorization is taken from manually prepared log files from the Phoenix system, developed by Dialogue Systems AB. This chapter consists of a brief overview over the parts in the system that is relevant for the understanding of this thesis. The text in this section is based on material from different handbooks for the Phoenix system.

### 3.1 System overview

The Phoenix system is a commercial natural language interface to databases. Phoenix is a portable system, in that it can be ported between different domains. A domain represents knowledge about a certain area. The system can work with different query languages. The components of the Phoenix system are a query interface and a natural language engine. The input to the query interface is a question in natural language. The natural language engine analyses natural language questions and generates an SQL statement used to retrieve data from a relational database.

#### 3.1.1 Schemes

The system consists of different conceptual schemes which together make a conceptual model. The conceptual model consists of the words that a user can use in the questions and it also describes how the words are related to each other and to the content and structure of the database. The conceptual model maps words and relationships in the natural language question to the tables and columns in the database and is used by the natural language engine to analyze natural language questions.

The conceptual schema is a kind of lexicon which includes information about object in the domain, such as database entities and grammatical relations between the entities. It also includes a concept hierarchy. An example of a lexicon entry is the verb “*samtala med någon*” (speak to somebody). It specifies a prepositional object relation between the entity for the verb and the entity for the prepositional object. (Lundstedt, 2002) The conceptual schema is a necessary component of a portable natural language interface to make it possible to use the system for different applications. This is because the information in the conceptual schema is needed to adapt the natural language engine to the domain that a particular user wants to ask questions about. There are three schemes that cover semantically different material. The user schema (application schema) consists of domain-specific information and a lexicon which is created during what within the Phoenix system is called the customization phase. A special customizing tool, which is a graphical editor, is used to develop the grammar and form the conceptual models which the application work with. The

base schema consists of the application-independent lexicon. There is also a meta schema which consists of concepts and vocabulary needed to talk about the Question Answering system itself.

### 3.1.2 Entity and term

A term, within the model, is a word in common case that can occur in one or several entities, i.e. can have any number of synonyms attached to it. The term represents the words that can be used in the natural language question. An entity within the model refers to a unique unit that consists of one or several synonymous terms. An entity has a unique name in the model and it belongs to a class or instance, which represents a general concept such as quantity, material thing and time. The class also partially decides which relation the entity can be entered into. An entity belongs to a category which is the same as part of speech of the entity and it can be a noun, an adjective, a verb and a proper name.

### 3.1.3 Lexicon

The system uses two lexicons. One is the basic grammar lexicon that is located in the base schema and which is used for all domain models. The other is the lexicon that is generated from the customization tool, which makes it domain specific and is located in the user schema. Both lexicons are represented as Prolog facts. The domain lexicon consists of several entrances which are connected to the information created in the customizing tool.

The following entrances are used in the categorization process in this study; *u\_syntax* describes the morphology information for a term. *u\_term*, consists of an identification number for the term and the common case form for the word. *u\_term* is the mapping between a term identifier and a term. A term can be a word or a structure representing a compound term. *u\_image* maps a term to an entity and *u\_category* consists of the term id and information about part of speech. *u\_dictionary* describes information about the term id, the word in common case and other lexical information.

### 3.1.4 The Phoenix parser

The semantic representation of a question is generated by a parser, along with a semantic analyzer. The analysis is done from the information about terms and relationships that are defined in the conceptual model. It also uses information from the base vocabulary and from the rules in the grammar. To produce a natural language interpretation and an SQL statement, a semantic connection between all the parts of the question needs to be found. If this is not the case, an error message is produced by the parser. The reasons why a question is not answered in the system can for example be that a question contains spelling errors or other grammatical errors. It can also be that the question consists of words that the system cannot understand or has a grammatical construction outside the master grammar specification, which defines what grammatical constructions the system can handle. An example of this is for instance temporal subordinate clauses. The result of this is that the question end up as non-answered in the log file.

The grammar formalism used in the Phoenix system is called XDLG, a formalism using standard Prolog notation, developed for grammar writing. Grammars written in this formalism are automatically translated into a DCG format. The output from syntactic analysis is one or more tree structures of semantic calls, so called semantic trees and these are used to generate the logical form in the following analysis module.

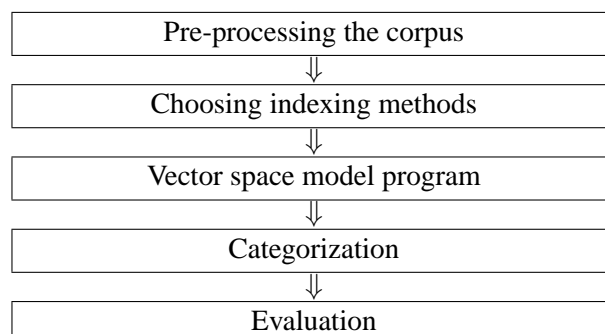
## 4 Experiments

This chapter describes the categorization process and the decision-making around the pre-processing of the data. The Java program, called the vector space model program, which is used to categorize and group the questions in the log files is also described in this chapter. This chapter also describes which features are used in the indexing process.

The purpose of categorizing the questions in the log files is to make it easier to find areas which the system cannot answer questions about yet. A sorted log file will facilitate the updating of the domain models. The application should therefore work in an environment where new paraphrase sets of questions are added and it should also be easy to apply to different domains. Another prerequisite is that the application should be able to handle the fact that the questions can belong to a large number of paraphrase sets, sometimes around 300 sets. A difficulty is that questions often consist of few words.

In a text categorization system, a document is assigned to a set of predefined document classes. In a document retrieval system, the user's query is used to find a suitable document. In this case, the material that is categorized is user queries, i.e. the non-answered questions in the log files. The query will be compared to a predefined paraphrase set of questions, consisting of different paraphrases of questions. This set can be seen as equivalent to documents. In this case the predefined paraphrase set correspond to a collection of such.

The comparison between a query and the paraphrase question-sets is done by using a vector space model. Two different kinds of features will be used for the indexing process, keywords expanded with synonyms and  $n$ -grams. An evaluation will be made in order to examine which of the index methods have the best performance.



**Figure 4.1:** Project outline.

There are two ways to select features from the non-answered questions. One way is to select features from the non-answered question to use in the index language is to

extract keywords from the question. The domain lexicon is then used to expand the keywords with their synonymous terms. The keywords and synonyms will be used as features for indexing the vectors. This feature extraction method makes use of the fact that synonymous relations for terms are available in the domain lexicon in the Question Answering system.

The second method for indexing is to extract  $n$ -grams from the questions and use these as features for indexing the vectors. Experiments were done using unigrams and bigrams of words and characters. Since the questions are domain specific, a high amount of domain specific collocations, such as “*avsluta förval*” (“*cancel account*”) and phrasal verbs such as “*säga upp*” (“*cancel*”), “*ringa ut*” (“*make a call*”) were expected. The hypothesis is that by using bigrams as features, i.e. using a bag-of-bigram representation, the semantics of the domain specific collocations can be captured and used in the categorization process.

The log files also consist of a number of questions about areas that are unknown to the Question Answering system. Since the content of these questions cannot be predicted, these questions cannot be categorized by comparing them to questions from the paraphrase set. For a categorization system, it is important for the categorization to be able to recognize also when a given query does not match any paraphrase set of questions. The purpose here is to categorize these questions as “unknown”, i.e. belonging to paraphrase set 0, (see 5.1).

## 4.1 Data

This section describes how the predefined paraphrase set of questions are made and secondly how the material of the log files has been pre-processed.

### 4.1.1 Predefined paraphrase sets of questions

For all domains in the Question Answering system there is a corpus with different framings of questions that will lead to the same answer in the system. The questions are manually collected and ordered in paraphrase sets. This corpus is used during the customization phase. The paraphrase sets in the corpus are used when constructing the paraphrase set for the categorization process in this thesis. This is an example from the question corpus for the domain Glocalnet:

*vilka plustjänster fungerar*  
*vilka plustjänster finns det*  
*vilka plustjänster har ni*

*vad är förval*  
*vad innebär förval*

The paraphrase sets in the Question Answering system corpus will correspond to paraphrase sets used in the categorization process. In this way the sets are easy to extract and will be equivalent to the already existing sets for a specific domain in the system.

The application was evaluated for two different domains. During the development phase, material from a domain called Glocalnet was used. The application is inten-

ded to be used by the staff at the Glocalnet customer service. When the categorization program was finished, material from a new domain, Svenska Ostindiska kompaniet, SOIC (The Swedish East India Company), was introduced. The application is available online, and visitors to the website use the application to ask questions about The Swedish East India Company.<sup>1</sup> For the Glocalnet domain, there is about 280 paraphrase sets in the corpus, but only a selection of them was used during the development and evaluation phase. For the Glocalnet domain, the predefined sets were collected from the top 100 questions answered in the system to cover a sample of questions. Altogether 69 paraphrase sets were collected from the top 100 questions.

For the second evaluation, all paraphrase sets in the corpus are used. The original plan was to use all available paraphrase sets, just as they are. When the material was studied, it was discovered that some sets consisted of information about nearly the same thing, but were divided into several groups. Some questions from those groups were tested in the Question Answering system and the questions lead to the same answering file in the system. Due to this, some changes to the paraphrase sets were made, in that some sets were merged into one. The number of paraphrase sets used for the second evaluations is 169.

Here are some examples of sets that were divided from the beginning, but was merged for the categorization process:

*Hur stora dieseltankar finns det? - Hur mycket diesel rymmer bränsletankarna?*  
(*How large are the diesel tanks - How much diesel does the tanks contain*)

*Vad kostar projektet? - Vad är beräknad kostnad för projektet?*  
(*What is the cost for the project - What is the estimated cost for the projet*)

*När startar resan? - När är den kommande resan?*  
(*When does the trip start - When is the coming trip*)

*vem blir befälhavare på båten? - vem är befälhavare på båten?*  
(*Who will be the captain of the ship - Who is the captain of the ship*)

A third small evaluation was also performed on the SOIC domain, using the same premise as for Glocalnet, regarding the paraphrase sets. The predefined sets were derived from the top 100 questions answered in the system, to cover a sample of questions. Altogether 61 paraphrase sets were derived from the top 100 questions for the SOIC domain.

#### 4.1.2 Pre-processing of the log files

There may be several reasons why a question is not answered in the Question Answering system and therefore ends up in the log files as non-answered. It might be due to grammatical errors, unclear formulations or that the question is about an area which is outside of the domain. Some of the questions to the Question Answering system are obviously outside the domain and can be considered as non-serious, for example “*Vem är du*” (“*Who are you*”) and “*Hur långt är det till stationen*” (“*How*

---

<sup>1</sup>The company has built a ship according to the dimensions to the hull and rigg of a Swedish East Indiaman that sank in the inlet of Gothenburg harbour in 1745

*far is it to the station*”) in an application about teleoperators. A delimitation of the material is that it should only consider serious questions and questions without grammatical errors. The first task in processing the log files is to filter out such questions from the log files. Due to limitations in time, the work of filtering out such questions from the log files were done manually.

Some spelling errors and abbreviations are included in the domain lexicon and these are not excluded from the log files. In those cases it is not due to the spelling error that the question has not been answered. A manual first sorting of the log files was done according to the following:

- Random inputs such as “*asdfv*” were erased.
- Obvious non-serious questions were removed.
- Questions with serious spelling errors and grammatical errors were removed.
- Questions containing abbreviations were removed.

The log files for the applications consisted of about 1400 questions per each and around 200 questions for each log file were removed in the first sorting. The tables 4.2 and 4.1 show example of questions from the log files.

## 4.2 Indexing methods

The comparison between the questions from the paraphrase set and the non-answered questions from the log file is made using a vector space model. Several kinds of features are used to index the vectors. The features are either keywords from the questions, expanded with synonyms from the domain lexicon or *n*-grams from the questions. The different term indexing methods used in the program will be described in the following sections.

### 4.2.1 Indexing with keywords and synonyms

The lexicon consists of information about how words are related to each other by the notion of term ids and entity id. The hypothesis is that it should be possible to find out how questions are related to each other by combining a keyword with its synonyms from the domain lexicon. The words that remain after the removal of stop words are considered to be keywords. This is a very simple way of extracting keywords. The stop words are removed both in the file with non-answered questions and in the file with the predefined paraphrase set.

A Prolog program was developed for the dictionary look-up. The program reads a file containing only keywords for either the non-answered questions or the questions from the paraphrase sets. Each question is tokenized and the term id for each keyword in a question is looked up in the dictionary. This is done in two different ways. For the Glocalnet domain, the dictionary entrance *u\_syntax* is used and for the SOIC domain, the entrance *u\_dictionary* is used. These two lexicon entries consists of all different forms of the word and the term id (see section 3.1.3). The difference between them is that *u\_syntax* consists of all forms of a word in one entry; *u\_syntax(t5, [pris, pris, priset, prisets, priser, prisers, priserna, prisernas, [c,n],i])*. and *u\_dictionary*

**Table 4.1:** Sample from the log file for the SOIC domain.

---

**Non-answered question from the SOIC domain.**

---

jag vill mönstra på till kina  
*I want to sign on to China*  
kan man medresa några dagar  
*is it possible to sail with the ship a few days*  
antal spikar som sitter i skrovet  
*number of nails in the hull*  
att söka ombordtjänst  
*apply for duties on board*  
besättningen  
*the crew*  
den kommande resan till kina  
*the coming trip to China*  
provseglingar  
*test sailings*  
segel  
*sails*  
hur gammalt är skeppet  
*how old is the ship*  
vad hände med den första ostindiefararen  
*what happened to the first East IndiaMan*  
hur många besöker ostindiefararen varje dag  
*how many visitors does the East IndiaMan have every day*  
hur många ostindiefarare fanns det  
*how many East IndiaMan were there*  
hände det nåt på vägen till kina  
*did something happened on the way to China*

---

consists of only one form in each entry;  $u\_dictionary(t5, nomen(nomen(490, feature(sg=1, indef=1, basic=1, specb=1, naked=1, count=1, neu=1, subj=1, obj=1, spec=1, ntr=1, lg=1, lp=1, lab=22), pris), [], nomen(t5))) => (str([112, 114, 105, 115]))$ ). Different entries have to be used because some of the words in  $u\_syntax$  in the SOIC domain lexicon were declared in English instead of Swedish. When the keyword from the log file which are in Swedish were looked up in the dictionary, it was not found since the English correspondence were saved in the dictionary entrance.

There is also another difference between the Glocalnet domain and the SOIC domain. In the lexicon for the SOIC domain some word forms are missing. Since there are common words as “*skeppet*”, (“*the ship*”) and “*segel*”, (“*sail*”) that are missing, those lexicon entries were added to the domain lexicon, otherwise this indexing method could not have been used.

Each term id is related to one or several entity ids and the next step for the program is to find all these entity ids and store them in a list. The output of the Prolog program consists of all entity ids for each question and is stored in a file. The file containing the entity lists is used as input to the vector space model program.

The following is a short description of the algorithm for the first indexing tech-

**Table 4.2:** Sample from the log file for the Glocalnet domain.

---

**Non-answered question from the Glocalnet domain.**

---

varför kommer jag inte ut på internet  
*why can't I access the Internet*  
varför har jag fått en faktura på 38 kr  
*Why have I got a bill at 38 kronors*  
kampanjer  
*campaigns*  
varför fungerar inte mitt adsl  
*why doesn't my adsl work*  
hur gör jag när jag byter adress  
*how do I change my address*  
hur mycket har jag ringt för  
*what is my phone call cost*  
uppsägning  
*cancellation*  
vad är glocalnet  
*what is glocalnet 0611*  
*0611 (area code)*  
jag vill åter ringa med er hur gör jag  
*I want to use your phone services again how do I do this*  
kan jag få 8mbit  
*can I have 8mbit*  
säga upp förval  
*cancel account*  
vad ligger öppningsavgiften på  
*what is the opening fee*

---

nique. The steps are made for both non-answered questions and for the questions from the paraphrase sets:

- Stop words are removed and the words that remain are considered to be keywords.
- Term ids for the keywords are looked up in the domain lexicon.
- Every entity synonym for the term ids is looked up in the lexicon.
- The features of the index language corresponds to the entities in the lexicon and categorization is done by comparing the entities from the log-question with the entities from the paraphrase sets by using the vector space model program.

#### 4.2.2 Indexing with $n$ -gram

Experiments have been done with both word unigrams and bigrams as index features. For bigrams, both word bigrams and character bigrams have been used as index terms. When bigrams are used as feature, the representation of the questions will be extended to a bag-of-bigram method (see 2.2). The word bigrams are made in order to match beginning of sentence and ending of sentence situations. The bigrams

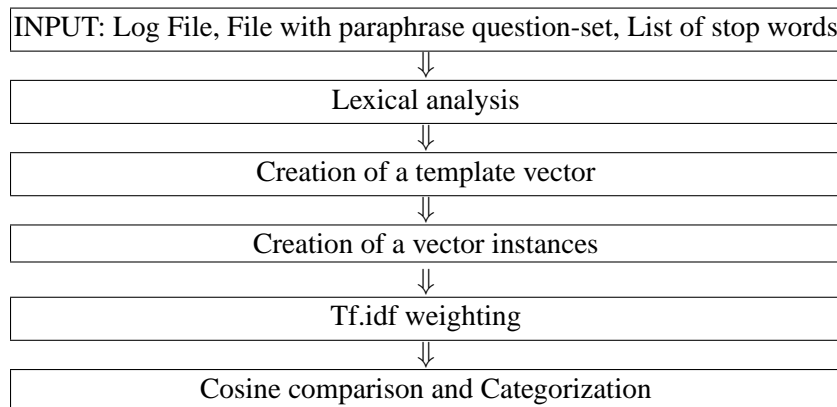
of characters are made in order to match beginning of word and ending of word. Therefore blanks have been appended to satisfy this. This is two examples of how the bigrams are formed, the underscore character represents blanks: “\_here, here is, is an, an example, example\_” and “\_e, ex, xa, am, mp, pl, le, e\_”.

All indexing methods based on  $n$ -grams are tested with stop words removed and stop words kept in the questions. In the vector space model program, there is a default setting which has to be changed in the cases where the feature bigrams made of characters are used. This is done by changing a variable in the beginning of the program.

### 4.3 The vector space model program

The comparison between the questions from the paraphrase set and the non-answered questions from the log file is made using a vector space model which is implemented in a Java program. Different types of features are used to index the vectors. To extract the synonyms for the keywords from the domain lexicon, a Prolog program is used (see 4.2.1). The output from the program is the input to the Java program which was developed to categorize the log files.

In the Java program, text normalization is performed, vectors for log-questions and questions from the paraphrase set are created and a comparison between the vectors by measuring the cosine between two vectors at a time is made. There are different settings for the program which states which type of feature to be used and also if stop words should be removed.



**Figure 4.2:** Steps in the vector space model program.

The Java program consists of six classes:

- *SortLog.java* - the main class.
- *VectorSpaceModel.java* and *VectorSpaceModelQQ* - methods performing the VectorSpaceModel calculations.
- *NgramMap.java* and *NgramMapQQ.java* - representation of all terms and their frequencies.
- *LogQuestion.java* - representation of a non-answered question and the assigned paraphrase number.

The input to the program is a log file, a file with paraphrase sets and a file with stop words if used. In the following sections the classes and the steps in figure 4.2 will be described further.

### 4.3.1 Lexical analysis

The original format of the log files is HTML and the questions are extracted from the log files and saved as text in the class SortLog. After that a lexical analysis is performed in which all tokens except letters and digits are removed. All capital letters are converted into lower-case. This is also performed by the class SortLog.

A removal of stop word can be performed for both non-answered questions and questions from the paraphrase set. The stop word list is used to filter out words from the questions that are not considered to be keywords. By filtering out stop words, the number of index terms is reduced. The stop word list consists of 180 Swedish words. Negations has not been considered as stop words since they might be regarded as relevant for the categorization process. The following example is taken from the Glocalnet domain, a domain covering questions about telephone problems: “*Jag vill inte ringa med er igen*”, (“*I don’t want to make calls with you again*”), “*Jag vill ringa med er igen*”, (“*I want to make calls with you again*”). These questions belong to different paraphrase set and the only difference is the negation. The stop word removal is optional and can automatically be made in the vector space model program if that option is chosen.

### 4.3.2 Creation of a template vector

Apart from the different indexing methods used in the categorization process, two different methods for representing the features of the paraphrase sets in vectors are used. The difference is that either is one vector created for all questions from a paraphrase set, or one vector for each question from a paraphrase set is created. The first vector representation aims to compare every question that belongs to a set, to a non-answered question. This method is referred to as *question-to-collection method*. The latter method aims to compare one question at the time from the sets to a non-answered question. The second method is referred to as *question-to-question method*. Thus, the difference for the vector representation is that for the question-to-collection method, one vector for all questions from the paraphrase set is created as opposed to one vector for every question from the paraphrase set for the question-to-question method.

The program consists of one class, VectorSpaceModel, which corresponds to question-to-collection comparison and another class, VectorSpaceModelQQ, which corresponds to question-to-question comparison. In both classes an array is created which eventually will contain all vectors for the questions from the paraphrase set.

The vectors consist of all terms in the document collection, with separate frequency calculations for each question. For the purpose of constructing a template vector, the classes NgramMap and NgramMapQQ are used. In the constructors of these classes, the file containing the paraphrase question-set are read. Depending on what type of indexing feature has been chosen, the NgramMap and NgramMapQQ constructors can for example create bigrams for the whole file and creates a template vector.

### 4.3.3 Creation of vector instances

In the class `SortLog`, the file with the paraphrase question-set is read, one line at the time. An empty line is interpreted as that the paraphrase set number should be increased. In `VectorSpaceModel`, an empty line is interpreted as “create a new vector” and “increase the set number”. In `VectorSpaceModelQQ`, an empty line is only interpreted as “increase the set number”. This is because in `VectorSpaceModelQQ`, a new vector is made whenever there is a new line.

A template vector is made for one question at the time and the next step in the program is to count frequencies of all the features in the template vector that was created by `NgramMap` or `NgramMapQQ`. A method in those classes, creates the index feature that is going to be used and calculates the term frequency for that feature. When this is done, the vector for the question is put in an array where all vectors for the paraphrase sets are collected. In this way, each vector can be used again, for the comparison with non-answered questions.

The next step in `VectorSpaceModel` is to read the file with all non-answered question. One line at the time is read and a template vector is created for every question. Then, the frequency of the terms for the non-answered question are calculated, in the specific instance of each vector. For the log-questions, the vectors are retrieved and put in the array, also to be used for the comparison with the vectors for the questions from the paraphrase set.

### 4.3.4 Tf.idf weighting of the terms

The *tf.idf* measure is based on values such as document frequencies. This means that the *tf.idf* weight can not be applied to the terms before the number of documents in the collection is known. Because of this, the simple term-frequency is re-weighted after all vectors is created. To calculate the *tf.idf*-value, different methods in the class `VectorSpaceModel` and `VectorSpaceModelQQ` are used. These are methods for calculating for example *df*, and *idf* (see 2.1.2). All terms in every vector are re-weighted with the *tf.idf*-weight.

### 4.3.5 Cosine comparison and categorization

The array containing all vectors for the non-answered questions and the array with all the vectors for the questions from the paraphrase set are iterated and normalized. Methods for normalization of the vectors are located in the class `NgramMap`. After the normalization, the dot-product of each vector for the non-answered questions and all vectors for the questions from the paraphrase set, one at the time, are calculated. The dot-product of two normalized vectors is defined as:

$$dot(d_j, d_k) = \vec{d}_j \bullet \vec{d}_k = \sum_{i=1}^N w_{i,k} \times w_{i,j} \quad (9)$$

When the highest dot-product between a vector for the non-answered question and a vector for a question from the paraphrase set is found, the question is assigned to that paraphrase set.

### 4.3.6 Final grouping of questions of the log files

After all non-answered questions have been categorized, the final step is to group the questions in the log file. All questions assigned into paraphrase set 1 is grouped together and so on. In the class `LogQuestion`, a representation of the questions is saved, as a question text and an assigned paraphrase set number. The class implements the interface `Comparable`, to be able to sort the questions after paraphrase set number. Appendix D shows a sample of a log file after categorization and grouping. The example is created when using unigrams with stop words kept as features, comparing question-to-collection. The example shows questions that has been categorized into paraphrase set 0.

## 5 Evaluation

The evaluation of the program has been done in two phases. The first evaluation was done on material from the Glocalnet domain during the middle of the development phase, as a test of how the different indexing techniques worked and to find out what further improvements could be done. The second evaluation was made on material from a new domain. This material was introduced during the last phase of the work. The domains used for evaluation have been chosen by Dialogue Technologies AB. The domain for the second evaluation is SOIC, *Svenska Ostindiska kompaniet*. The reason for choosing a new domain for the second evaluation was to see if the text categorization program performs equally for different domains. Each of the different indexing techniques have been evaluated and every technique is evaluated on question-to-question and as question-to-collection.

To make the evaluation possible, the output from the program during the evaluation step was the questions from the log file with a paraphrase set number attached to them. The final output from the system is a categorized log file, with questions belonging to the same paraphrase set grouped together. The assigned set number was compared to the key. This is an example of the output:

*Log question nr 1 matches Category nr 8 with cosine 0.9999999999999999*  
*Log question nr 2 matches Category nr 8 with cosine 0.8333333333333336*  
*Log question nr 3 matches Category nr 23 with cosine 0.8944271909999159*

### 5.1 Manual categorization of a key

The first evaluation were made on the first 150 questions (out of 1200) in the log file. A key for the first 150 questions was made by manually assigning each of them to one or several paraphrase sets. This was done by the author. In the first evaluation the system was tested on a selection of 69 of the total 280 sets (see section 4.1.1).

**Table 5.1:** Features used for indexing.

<b>type of feature</b>	<b>stop words</b>
Unigram of words	yes/no
Bigram of words	yes/no
Entities	no
Bigram of characters	yes/no

The second evaluation was performed on questions from the SOIC domain. 200 questions were randomly selected from the log files from a total of about 1200 questions. Those questions were also manually assigned to one or several sets. These two steps were performed by Joakim Ingers at Dialogue Technologies AB. In this evaluation the system used all available paraphrase sets for the domain which was 169 sets.

For the third smaller evaluation of the SOIC domain, the same 200 questions as for the other evaluation was used. Since the paraphrase sets were changed, the key was modified by the author. This evaluation was only performed for one indexing method, using unigrams without stop words as features.

For some of the questions, it is difficult to decide which set they belong to. For example “Vem betalar”, (“Who is paying”) could refer to both paraphrase set “Hur finansieras bygget”(“How is the construction financed”) and “Vem sponsrar” (“What are the sponsors”). In those cases the key states that both sets are correct.

Some of the questions do not belong to any paraphrase set. It was decided that those questions should be assigned to a paraphrase set called 0. When the categorization program assigns a question to paraphrase set 0 and the key also states 0, this is counted as a correct categorization. A question that has been assigned to paraphrase set 0, although there is another set that is more appropriate, has been counted as an error.

The distribution of number of questions to each paraphrase question-set vary some. Some paraphrase sets are more common than others. Tables 5.2 and 5.3 show how many of the manually categorized questions belong to each set. The sets which have the highest number of question assigned to it are in both tables paraphrase set 0. 24 sets for Glocalnet and 91 sets for SOIC has not any question assigned to it. There are different explanations to this. One explanation is that the key consists of 150 respectively 200 questions from a log file of 1400 questions. Since the questions have not been chosen to cover all different kinds of areas, some paraphrase sets are not covered in the key. Another reason is that some questions are more unusual than others. It can also be due to that all questions about that paraphrase set are answered by the Question Answering system.

For the Glocalnet domain, 21% of the 150 evaluated questions were manually assigned to paraphrase set 0. For Glocalnet, only 69 of the around 280 paraphrase sets available are used. Due to this, there might be a number of questions that will fall outside the scope of the sets used. There will always be a certain number of such questions, but in this case it might be a higher rate.

## 5.2 Evaluation measures

The system will be evaluated by measuring accuracy, precision and recall. Accuracy is defined as the ratio of correctly assigned items over the total of items. Items are in this thesis corresponding to questions categorized by the program.

The percent of how many questions that have been assigned to paraphrase set 0 will also be calculated. The precision for the questions assigned to paraphrase set 0 will also be calculated.

An evaluation of the paraphrase set of questions has also been made. Table 5.2 and 5.3 shows the distribution of assigned question to each set. The evaluation of the sets has been made for *unigrams with stop words kept* and *bigrams of characters*

**Table 5.2:** Question distribution over paraphrase sets in the key for the Glocalnet domain, total number of paraphrase sets is 158, total number of questions is 150.

Paraphrase sets	% of the questions
1	22
1	9
2	5
1	4
8	3
6	4
8	2
8	1
17	0.7
24	0

**Table 5.3:** Question distribution over paraphrase sets in the key for the SOIC domain, total number of paraphrase sets is 244, total number of questions is 200.

Paraphrase set	% of the questions
1	40
1	12
2	5
1	3
15	2
8	1
48	0.5
91	0

with stop words kept. Precision and recall has been computed for the sets that have 3% of the questions assigned to it. Precision and recall are for each set measured as:

$$\text{RECALL} = \text{questions found and correct} / \text{total number of correct question}$$

$$\text{PRECISION} = \text{questions found and correct} / \text{total question found}$$

The measures precision and recall are used to evaluate for example information retrieval systems with respect to the notion of relevance. For categorization there could be interesting to consider both sides, what is called relevant and non-relevant classes. When the measure accuracy is used, the degree in which the system does not retrieve irrelevant instances is also taken into consideration. Accuracy and recall are similar but the difference is that accuracy use both the relevant and the non-relevant class together.

### 5.3 Results

The results for the Glocalnet domain shows an average accuracy rate of 53% for the question-to-collection comparison, with an accuracy rate of 59% for the best performing index method and an accuracy rate of 46% for the lowest performing

index method. For the question-to-question comparison the average accuracy rate is higher, 58%, where the highest and the lowest performing index methods had, respectively, 62% and 49% as accuracy rate. For the Glocalnet domain, 21% of 150 questions were manually assigned to paraphrase set 0.

For the question-to-collection comparison, the index method using features made of *bigram of characters with stop words* kept had the highest accuracy rate. Almost the same result was achieved when the method were *unigram with and without stop words* was used. This goes for *keywords expanded with synonyms* as well. The method were *bigram of words, with and without stop words* were used as feature had the lowest accuracy rate.

For question-to-question comparison, the highest accuracy rate was achieved for the method using *unigrams without stop words* as features. The index method using *bigrams of characters, with and without stop words* and *bigrams of words, without stop words* gave almost the same result. The accuracy rate was quite even for all the index methods, except for *bigrams of words with stop words* kept which had an accuracy rate that was lower than the others.

The clearest difference between the two ways of comparing the questions are that indexing with *bigrams of words without stop words* performs very well in question-to-question comparison but has the lowest accuracy rate of all index methods in question-to-collection.

The results for the SOIC domains shows an average accuracy rate at 22% for question-to-collection comparison and an average accuracy rate at 26% for question-to-question comparison. The highest performing index method for the question-to-collection comparison had an accuracy rate at 27% and the lowest performing index method had an accuracy rate at 20%. For the question-to-collection comparison the highest performing index method had an accuracy rate at 29% and the lowest performing index method had an accuracy rate at 21%. For the SOIC domain, 32% of 200 questions were manually assigned to paraphrase set 0.

The best result for question-to-collections is for *bigrams of words without stop words*, closely followed by *bigrams of words with stop words*. *Bigrams of characters without stop words* has the lowest accuracy of all the index methods.

The index method which has the highest accuracy rate for SOIC question-to-collection comparison, also has the highest accuracy rate for question-to-question comparison, together with using features of *unigram without stop words*. Almost every index method has the same result, except when using the feature *bigram of characters without stop words*, which performs lower than the others.

The index methods with the highest accuracy for the Glocalnet domain are all applicable to the SOIC domain with one exception, *bigrams of characters without stop words* which does not perform very well. The index method *bigrams of words without stop words* has the highest performance for SOIC and it also performs very well for Glocalnet question-to-question, but lower for question-to-collection.

The evaluation for the SOIC domain shows that when a smaller number of paraphrase sets were used, the program performed much better than when a larger number of sets were used. The accuracy was improved from 29% accuracy to 40%. 22% of the questions were assigned to paraphrase set 0. The precision for questions categorized into paraphrase set 0 was also higher than it was when more paraphrase sets were used, from 55% accuracy to 63% accuracy.

A small evaluation has also been made for the paraphrase sets. Appendix A shows an evaluation of precision and recall over the sets for the Glocalnet domain when the

**Table 5.4:** Results for the Glocalnet domain, question-to-collection comparison.

<b>Index feature</b>	<b>Accuracy</b>	<b>%, set 0</b>	<b>Precision, set 0</b>
Unigram, with stop words	56	17	62
Unigram, without stop words	56	23	51
Bigram, words/with stop words	45	23	49
Bigram, words/without stop words	46	35	45
Bigram, characters/with stop words	59	0	-
Bigram, characters/without stop words	51	0.7	0
Entities, without stop words	56	19	59

**Table 5.5:** Results for the Glocalnet domain, question-to-question comparison.

<b>Index feature</b>	<b>Accuracy</b>	<b>%, set 0</b>	<b>Precision, set 0</b>
Unigram, with stop words	56	17	62
Unigram, without stop words	62	21	56
Bigram, words/with stop words	49	20	50
Bigram, words/without stop words	61	24	50
Bigram, characters/with stop words	60	0	-
Bigram, characters/without stop words	59	0	-
Entities, without stop words	58	17	56

**Table 5.6:** Results for the SOIC domain, question-to-collection comparison.

<b>Index feature</b>	<b>Accuracy</b>	<b>%, set 0</b>	<b>Precision, set 0</b>
Unigram, with stop words	20	8	50
Unigram, without stop words	21	17	55
Bigram, words/with stop words	24	22	49
Bigram, words/without stop words	27	40	50
Bigram, characters/with stop words	20	0	-
Bigram, characters/without stop words	20	0	-
Entities, without stop words	23	29	45

**Table 5.7:** Results for the SOIC domain, question-to-question comparison.

<b>Index feature</b>	<b>Accuracy</b>	<b>%, set 0</b>	<b>Precision, set 0</b>
Unigram, with stop words	26	7	38
Unigram, without stop words	29	16	48
Bigram, words/with stop words	28	13	50
Bigram, words/without stop words	29	24	54
Bigram, characters/with stop words	26	0	-
Bigram, characters/without stop words	21	0	-
Entities, without stop words	21	29	46

index method *bigram of characters with stop words* and *unigram with stop words* are used. The results show an even distribution between either higher recall or higher precision for the categories. For three of ten paraphrase sets, the recall rate was under 0.5. These paraphrase sets were concerning prices, how a customer can change his mind about an order and changing of address. These questions have instead been assigned to adjacent paraphrase sets or to paraphrase set 0. For example, a question about changing address has been categorized to the set concerning changing telephone number. There are several paraphrase sets concerning questions about prices. The question “pris” (“price”), are categorized into the paraphrase set concerning the price for calling voice mail and the key states this as an error. This is an example of that the key might have been different if another person had made the manual categorization. Three of ten paraphrase sets had a low precision rate. The most common thing among those sets is that questions that should have been categorized into paraphrase set 0 has been categorized into those three. In those cases the question consists of one or several common words, although the question is not a paraphrase.

## 6 Conclusions

The purpose of this thesis was to find a way to automatically assign questions to pre-defined paraphrase sets of questions. The categorization was done by writing a Java program in which a vector space model was implemented. The input to the program is a log file consisting of non-answered questions from the Question Answering system. The output is the questions from the log file, grouped into paraphrase sets. The set number can be attached to the question if wished.

The Java program presents a general solution for categorizing the questions in the log file. The input of the log files is the HTML format that Dialogue Technologies AB use. The HTML is then converted to text, different text normalization steps are performed on the files and a vector comparison is done. This is done automatically in the program. The user can choose which type of index feature to use each time the program is run. If the user wishes to use the index feature keywords with synonym expanded, the input to the program has to be the lists of entities from the Prolog program, for both non-answered questions and question from the paraphrase set.

An evaluation was done for two different domains, Glocalnet and SOIC. For the Glocalnet domain, 69 different paraphrase sets were used in the categorization process. The program had an average accuracy of 56%, 62% accuracy for the best performing indexing method. For the SOIC domain, 169 paraphrase sets were used and the average accuracy was 24%, 29% accuracy for the best performing indexing method. For both domains, the best results were accomplished when the categorization was made by comparing question-to-question, as opposed to question-to-collection.

There are explanations to the poor results for the SOIC domains. 40% of the questions for the SOIC domain belonged to paraphrase set 0. Many of these questions consists of words such as “fartyg” (“ship”) and “besättning” (“crew”), the same words as for questions which belong to a predefined paraphrase set. In those cases the vector space model finds, as it should, similarities between the vectors, although the cosine rate is low. The lower the cosine, the higher the probability that the decision to assign a question to a paraphrase set is wrong. A solution to this problem is to add a threshold for the similarity comparison to the program. For example, if the similarities between two vectors are lower than 0.4, the question is assigned to paraphrase set 0. Some correct categorizations might be missed but it might improve the accuracy of the program.

The second explanation to the poor results is that the first evaluation of the SOIC domain probably suffers from comparing with too many paraphrase sets and too narrow dividing of the sets. 169 paraphrase sets were used, in comparison to 69 for Glocalnet. The second evaluation for the SOIC domain, where only 61 paraphrase sets were used performed much better. The second evaluation for the SOIC domain, together with the evaluation for the Glocalnet domain, support the hypothesis that the

categorization program performs better when a smaller number of paraphrase set are used.

The accuracy rate for paraphrase set 0 was also higher when fewer paraphrase sets were used. When fewer sets were used, the accuracy rate increased from 55% accuracy as the best result, to 63% accuracy.

For the different indexing methods, the feature *bigrams of characters* will almost always have some bigram in common with a question from the paraphrase sets. When using *bigrams of characters* as indexing features, a question will almost never be assigned paraphrase set 0. This is a disadvantage for this indexing method but despite this the indexing method performs very well. When this indexing technique is used, most of the wrong decisions for the categorization program are cases where the question should have been categorized to paraphrase set zero, otherwise that indexing method rarely makes mistakes. To combine *bigrams of characters* as feature with a threshold in the program would be interesting to examine further.

The number of questions that belong to paraphrase set 0 also affected the result for the SOIC domain. Since there where a high number of such questions, the conditions for indexing techniques such as *bigram of characters* are not very good.

A tendency for the Glocalnet domain is that the accuracy rate is higher for the index methods where stop words are removed. The removal of stop words also gives better result for the SOIC domain. The only exception to this, for both domains, is when *bigrams of characters* are used as index method. For that index method it is better to keep the stop words. The differences in accuracy rate are very small though, so it is difficult to draw any conclusions from this.

Categorization using *entities* from the domain lexicon as features performed well while comparing question-to-question. This method could be extended to extracting phrases instead of just single words since this information is available in the lexicon for the Question Answering system. This could be a possible way to enrich the bag-of-words method, similar to the bag-of-bigram method.

The final grouping of the log files will be a little better than the figures show. This is because similar paraphrase sets are placed after each other in the file. The evaluation of the paraphrase sets shows that the system sometimes has trouble distinguishing between two similar sets. This is because the sets have several common words. Using another weighting schema might affect this results. If similar paraphrase sets follows after each other in the file, the categorization will be wrong by one or two numbers. In those cases the questions will be grouped together anyway in the output, and the patterns in the logfile can be more easily found.

No stemming or lemmatization was performed on the keywords. The use of stemming or lemmatization could possibly improve the result.

In the beginning when this application is supposed to be used, the log files will still consist of several spelling errors. In such an environment, using *bigrams made of characters* as feature might be useful, since bigrams of characters are less sensitive to spelling errors than unigrams or bigrams of words.

A log file where the non-answered questions are collected into paraphrase sets facilitates the upgrading of the domain. One important thing is to find questions referring to new areas and wich has no correspondance in the paraphrase set, i.e. question that should be categorized to paraphrase set 0. The percent of questions assigned to paraphrase set 0 are quite equal for all indexing methods, except when using *bigrams of characters* as features, which almost never assigns a question to set 0. Because

of this, it might be preferable to use another indexing method, for example *unigrams without stopwords*, although *bigrams of characters* has a high accuracy rate.

There is also a possibility that adding a threshold to the program for the similarity comparison of the vectors could improve the result for several indexing features, not only *bigrams of characters*. A future development of the vector space model program could be to try to group the questions in paraphrase set 0 into new sense categories.

# Bibliography

- J. Allan and R Bekkerman. *Using Bigrams in Text Categorization*. Departement of Computer Science, University of Masschusetts, 2003.
- W. Cavnar and J. Trenkle. *N-Gram-Based Text Categorization*. Proceedings of SDAIR, 3rd annual Symposium on Document Analysis and Information Retrieval, 1994.
- A. El-Nasan, S. Veeramachaneni, and G. Nagy. *Word Discrimination Based on Bi-gram Co-occurrences*. Doclab, Resselear PolyTechnic Institute, Troy NY, 2001.
- A. Hulth. *Improved Automatic Keyword Extraction Given More Linguistic Knowledge*. Departement of Computer and System Science, Stockholm, 2003.
- D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, New Jersey, 2000.
- J. Karlgren. *Information Retrieval*. Swedish Institute of Computer Science, Stockholm, 2000.
- K. Krüger-Thielmann and H. Pajmans. *Information Retrieval*. Virtugrade information retrieval course, <http://pi0959.kub.nl/Paai/Onderw/Vir>, Last visited in March 2005, 2005.
- H. Kyoung-Soo and C. Hoojung. *Korea University Question Answering System at TREC 2004*. Conference Notebook Papers, TREC-6, 2004-13.
- K. Lundstedt. A computational grammar for swedish noun phrases in a natural language interface to databases. Master's thesis, Uppsala University, 2002.
- C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 2002.
- J. Mayfield and P. McNamee. *N-gram vs. words as indexing term*. Conference Notebook Papers, TREC-6, 1997.
- F. Sebastiani. *Machine Learning in automated text categorisation: a survey*. Istituto di elaborazione dell'Informazione, Pisa, IT, 1999.
- Y. Yang. *An Evaluation of Statistical Approaches to Text Categorization*. Information Retrieval, 1(1/2):69-90, 1999.
- B. Yatez, R. Neto, and B. Ribeiro. *Modern Information Retrieval*. Addison-Wesley-Longman, Harlow, U.K., 1999.

## A Evaluation of paraphrase sets, Glocalnet.

**Table A.1:** Glocalnet: Distribution of number of question to the paraphrase sets with at least 3 % of the question assigned to them. The table shows precision and recall for index method using *unigrams with stop words* and *bigrams of characters with stop words* as features.

Paraph. set	% of Q	Recall, 1-gram	Precision, 1-gram	Recall, 2-gram	Precision, 2-gram
0	21	0.48	0.62	0.03	1
1	3	0	-	0.25	1
8	9	0.85	0.81	1	0.81
12	5	0.38	0.75	0.75	0.86
27	4	0.83	0.71	0.67	1
28	5	0.86	1	0.86	0.67
42	3	0.75	0.6	0.5	0.4
54	3	0.5	0.66	1	1
61	3	0.6	0.38	1	0.63
63	3	0.2	0.5	0.6	1

## B Distribution of numbers of questions to each paraphrase sets for the Glocalnet.

The first number is the paraphrase set number, the second number is the number of questions assigned to that paraphrase set.

0:33, 1:4, 2:2, 3:3, 4:0, 5:2, 6:3, 7:1, 8:13, 9:1, 10:0, 11:3, 12:8, 13:3, 14:2, 15:0, 16:0, 17:0, 18:0, 19:0, 20:0, 21:1, 22:0, 23:0, 24:0, 25:0, 26:1, 27:6, 28:7, 29:1, 30:3, 31:2, 32:0, 33:1, 34:1, 35:2, 36:1, 37:2, 38:0, 40:1, 41:0, 42:4, 43:3, 44:4, 45:1, 46:4, 47:1, 48:3, 49:1, 50:0, 51:0, 52:1, 53:0, 54:4, 55:1, 56:0, 57:1, 58:3, 59:0, 60:0, 61:5, 62:0, 63:5, 64:1, 65:2, 66:0, 67:0, 68:4, 69:1

## C Distribution of numbers of questions to each paraphrase sets for the SOIC domain.

The first number is the paraphrase set number, the second number is the number of questions assigned to that paraphrase set.

1:1, 2:1, 3:0, 4:1, 5:0, 6:1, 7:0, 8:3, 9:0, 10:0, 11:0, 12:2, 13:5, 14:0, 15:3, 16:0, 17:9, 18:0, 19:0, 20:0, 21:0, 22:0, 23:0, 24:0, 25:0, 26:0, 27:0, 28:0, 29:4, 30:0, 31:2, 32:2, 33:1, 34:0, 35:0, 36:0, 37:1, 38:1, 39:0, 40:0, 41:0, 42:0, 43:0, 44:0, 45:1, 46:0, 47:0, 48:0, 49:0, 50:0, 51:0, 52:2, 53:2, 54:1, 55:1, 56:1, 57:1, 58:0, 59:0, 60:0, 61:0, 62:0, 63:0, 64:0, 65:0, 66:0, 67:0, 68:0, 69:0, 70:1, 71:0, 72:0, 73:0, 74:0, 75:0, 76:0, 77:1, 78:0, 79:0, 80:0, 81:1, 82:4, 83:1, 84:3, 85:3, 86:3, 87:1, 88:1, 89:1, 90:1, 91:1, 92:1, 93:1, 94:2, 95:0, 96:0, 97:0, 98:0, 99:0, 100:0, 101:0, 102:0, 103:0, 104:0, 105:1, 106:1, 107:1, 108:0, 109:0, 110:0, 111:0, 112:0, 113:3, 114:0, 115:1, 116:3, 117:1, 118:0, 119:7, 120:0, 121:0, 122:0, 123:0, 124:2, 125:0, 126:0, 127:0, 128:0, 129:0, 130:0, 131:0, 132:1, 133:1, 134:0, 135:4, 136:0, 137:1, 138:1, 139:0, 140:0, 141:3, 142:3, 143:10, 144:4, 145:24, 146:8, 147:4, 148:1, 149:0, 150:1, 151:0, 152:2, 153:4, 154:3, 155:1, 156:0, 157:1, 158:1, 159:1, 160:1, 161:1, 162:1, 163:1, 164:1, 165:1, 166:1, 167:1, 168:1, 169:1, 170:81

## D Sample from categorized log file, questions assigned to paraphrase set 0.

The example is from using unigrams with stopwords kept as features, comparing question-to-collection. The example shows which questions that has been categorized into paraphrase set 0.

```
0611 -> 0
chef -> 0
utfors -> 0
viktiga personer -> 0
villaägarna -> 0
ångerkonto -> 0
årsfaktura -> 0
0200_info_bilaga -> 0
04csr_8tel -> 0
04www_gn1 -> 0
ansvariga -> 0
gällande ångerkontot -> 0
ikea -> 0
ikeaanställd -> 0
ljudaccept -> 0
nummerpresentation -> 0
reklamation -> 0
ringring -> 0
salesorigin: 0200_info_bilaga -> 0
signal -> 0
startpaket -> 0
telenordia -> 0
tt -> 0
utfors -> 0
prislistan -> 0
utlands -> 0
vd -> 0
wal -> 0
ånger -> 0
021 0309_tipsgn -> 0
0404_at_fb-d -> 0
0406www_gbg5tel -> 0
0408adr -> 0
```

0410adr\_gn\_tel -> 0  
0410adr\_fb\_8tel -> 0  
0411 -> 0  
0457 -> 0  
04\_web\_vä4 -> 0  
04csr\_5tel -> 0  
04csr\_8 -> 0  
04www\_gn3 -> 0  
060 -> 0  
0977 -> 0  
1.3 -> 0  
13 -> 0  
148014 -> 0  
2569012 -> 0  
45 -> 0  
4787204-9 -> 0  
5161 -> 0  
5612985 -> 0  
7hu -> 0  
841 -> 0  
85 -> 0  
9545 -> 0  
ansvarig -> 0  
antichurn -> 0  
bassurf helkund -> 0  
bestridan -> 0  
customer care -> 0  
cv -> 0  
där borta ; vem äger dom -> 0  
euro\_0026 -> 0  
external order number 155211692164 not accepted. error message: delivery time ->  
0  
is longer than requested -> 0  
fakturaspecefikation -> 0  
fel fakturering -> 0  
fel ringning -> 0  
felfakturering -> 0  
abonnamanget -> 0  
frågar efter vårt faxnummer -> 0  
gprs -> 0  
gtel -> 0  
gtel flagga -> 0  
gtelefoni' -> 0  
helkund -> 0  
ikea anställd -> 0  
ikea anställda -> 0  
ikea familly -> 0  
ikea family -> 0  
ikea-anställd -> 0

inkommande epost -> 0  
inportering -> 0  
instruktioner -> 0  
ip -> 0  
isdn -> 0  
snarast -> 0  
kundnummer -> 0  
där -> 0  
använts -> 0  
adslinställningarna -> 0  
operatör -> 0  
försvunnit -> 0  
kundservicechef -> 0  
landskoder -> 0  
lön -> 0  
merförsäljning -> 0  
mobilabbonemang -> 0  
mobilpriser -> 0  
mobilsvar -> 0  
mobilt -> 0  
modem -> 0  
nedladdning -> 0  
nedladdningshastighet -> 0  
nummerupplysning -> 0  
porteringsavgift -> 0  
prislistamobil -> 0  
promocodes -> 0  
ranger\_surf -> 0  
retur -> 0  
ringringalltid -> 0  
roamingpartner -> 0  
roaming -> 0  
sales orgin -> 0  
sales orgin 04csr\_5tel -> 0  
sales orgin04csr\_5tel -> 0  
salesorigin -> 0  
salesorigin 04www\_gn1 -> 0  
salesorigin 04www\_gn3 -> 0  
saudarabien -> 0  
leverantör -> 0  
smtp -> 0  
sportster flash -> 0  
status ändring -> 0  
statusändring -> 0  
stämmer priserna -> 0  
surf plus -> 0  
syrien -> 0  
säkerhetspaket -> 0  
te3st -> 0

telefonsvarare pris -> 0  
telenor -> 0  
tm\_li\_29\_12m\_chrun -> 0  
ungdom -> 0  
unviersal -> 0  
uppladdnings hastigheten -> 0  
uppstörms -> 0  
ut\_bassurf helkund -> 0  
utfors helkund -> 0  
utfors surf -> 0  
utlandspriser telenordia -> 0  
telefonbolagen -> 0  
billigare -> 0  
telefonnumret -> 0  
operatör -> 0  
byte telefonnummer -> 0  
ang. detta) -> 0  
vidarekoppling -> 0  
villa ägarna -> 0  
villaägare §1.3 -> 0  
ångerkontot -> 0  
återbetalning -> 0  
öppnings avgifter -> 0