Master of Science Thesis

In response to your inquiry

Automatic E-mail Answer Suggestion
in a Dutch Contact Centre

Michel Ronald Boedeltje

October 2005

Human Media Interaction Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
Enschede
the Netherlands

In collaboration with

Em@ilco.nl B.V.
Amersfoort
the Netherlands

Abstract

In the past years, the number of service requests through e-mail has shown an explosive growth. To cope with the increased numbers of received e-mails, current call centres are "transformed" into so-called contact centres, handling both e-mail and telephone calls. Personal answering (each e-mail is written again by an agent) costs way too much, so most companies use a set of predefined answers that cover most of the e-mail service requests. Incoming e-mails are analysed, where after a set of (maximal) 10 possible answers is presented to the agent. If the selection is successful, writing an answer is replaced by selecting the correct (predefined) answer. Because this selection process works much faster, the number of e-mails that can be handled increases significantly. This is true, as long as the correct answer is presented within the set of suggested answers. If not, the agent has to find the correct answer by typing in those keywords that lead to the right set of suggestions: a time and money consuming process.

Therefore, the focus of this study is on the "improvement of relevant answer suggestions". We try to tackle the answer suggestion problem by transforming the incoming e-mail (the raw question) into a normalized question (uniform way of writing, removal of non-information words, etc.). This normalized question will be classified as belonging to one or more classes (each class has one predefined answer). In this thesis we present two classification techniques (known from regular text classification problems) and apply them to a corpus of 17,000 e-mails collected and classified in a Dutch contact centre. The classification techniques are adjusted in such a way that the implemented application can present a ranked list of relevant categories (representing the answer suggestions) for each incoming e-mail. We will show that both classification techniques are very suitable for automatic answer suggestion in a contact centre: our best method is able to present the correct answer within a ranked list of 5 possible answers, for almost 85% of all incoming e-mails.

Moreover, we will show that applying language technology as an advanced text normalization step, may improve the classification accuracy with another 3 percentage points. This brings the final classification accuracy to approximately 88%, which more than doubles the performance of the keyword based system as currently used by the principal of this research: Em@ilco. Finally, we will discuss the possibilities of completely auto-answering incoming e-mails: the ultimate dream of each contact centre manager. Although the results are not "good enough", for unsupervised auto-answering, our 60%-first-answer-right-result is quite promising. If combined with reliable confidence scores (something out of the scope of this research), the proposed methods can be used to automatically answer incoming e-mails (at least in out-of-office hours).

Preface

This thesis describes the research I have performed for my graduation project for the HMI group of the University of Twente and Em@ilco.nl B.V., in order to complete my Computer Science education at the faculty of Electrical Engineering, Mathematics and Computer Science. Off course, I could not have done all this work alone. Therefore, I gratefully thank the following people:

First, the people from Em@ilco, and especially Leo van Veen and Berry Cornelisse for giving me the opportunity to carry out my graduation project for Em@ilco. They have always helped me when I needed new data for my experiments or good programming advice to implement the prototype system.

Danny Lie from Carp technologies, for making available the Lingware tool-kit to help me investigate the use of specific language technologies in this research.

My graduation committee: Arjan van Hessen, Theo Huibers, Franciska de Jong and Leo van Veen. A special thank you to Arjan van Hessen, for being very enthusiastic about my research, for keeping me motivated, for the good advice and the nice conversations.

Off course I owe it to my fellow students of the Black-Coffee room to mention them in this section, for the fun and discussions of news articles, formula 1 races and our graduation work, during the last months.

My parents (Jan en Mineke) for enabling me to complete my education, their support and their continuous interest.

My little sister (Marion), room mates and friends (especially Laurens for reviewing this thesis), for their interest and support during this project.

Finally, I owe much gratitude to my girlfriend Jantina for her unconditional support and interest, but moreover, for the great times we had when I took some time off this project.

Michel

Contents

Chapter 1

Introduction

With the ongoing acceptance of e-mail as a fast, cheap and reliable means of communication, companies receive an increasing number of service requests via e-mail. To handle these e-mails, current call centres are “transformed” into contact centres; handling e-mail, telephone calls and, probably in the near future, chats as well. Since most service requests are about a relatively small set of problems, lots of these requests may be answered using a relatively small set of standard answers.

1.1 Problem description

Handling such great amounts of e-mail in a contact centre is a very labour-intensive task, requiring a serious investment of time and money. Automatizing the process of answering the service requests through e-mail could therefore account for a time and money reduction. In an ideal situation, a computer application would be developed that automatically selects the correct answer to an incoming service request and sends the reply to the customer without intervention of an agent. Unfortunately, the process of automatically answering e-mail is a very difficult task, making it very unlikely that the correct answer is selected for all incoming messages (recent studies show an automatic answering ratio of e-mail in which 40% of the mails is answered correctly,see chapter 2). Due to the risk of sending the incorrect answer to a customer, most companies are reluctant to incorporate automatic e-mail answering in their contact centres. However, automatically suggesting relevant answers to incoming messages provides a good alternative. If the correct answer to a question is presented within, for instance, a top-5 of relevant answers, the agent only needs to select the correct answer and send it to the customer. Such a system would improve the efficiency in a contact centre and reduce the time spent on answering e-mail.

The focus of this project is to investigate the possibility to semi-automate the answering of incoming e-mail by suggesting relevant answers to incoming messages.

1.2 Problem statement

Em@ilco has developed an e-mail management system (Q.mail//Box) that enables contact centre agents to handle incoming messages efficiently. This system also provides functionality to automatically suggest relevant answers. This answer suggestion routine maps incoming e-mail to standard question, based on the presence of predefined keywords in the incoming message. The standard question is linked to a standard answer. A set of keywords is manually assigned to each standard question and the standard question that has the most keywords in common with the incoming e-mail, links to the best answer suggestion. The main goal of this project is to investigate to what extent new techniques improve the automatic suggestion of answers.

We try to tackle the automatic answer suggestion problem by transforming it into a text classification problem. The e-mail messages are the documents that should be classified and the classes in which they should be classified are the representations of the standard questions. If an e-mail is classified (i.e. mapped to a standard question), we can simply suggest the answer that is associated with the standard question representing the category. The classification of new messages is done by determining the similarity between the new messages and previously answered messages. Based on the assumption that similar questions require similar answers, the new message can now be categorized in the category that stores previously answered messages that are most similar to the new message. Finding relevant documents based on document similarity is a basic Information Retrieval task where relevant documents are retrieved by determining the similarity between a document and a user defined query. We state that automatically determining the similarity between new messages and previously answered messages using information retrieval techniques, outperforms the basic classification approach used by Em@ilco, leading to our first research hypothesis:

1. Information retrieval based classification of e-mail messages for automatic answer suggestion outperforms the manually defined keyword approach and is usable in a Dutch contact centre environment.

Since e-mail is such an easily accessible means of communication, e-mails often are not well formed documents: They may contain spelling errors and grammatically incorrect sentences, which may negatively influence the performance of the classification algorithms. To overcome this problem, basic language technology may be used to normalize the e-mail before they are used in the classification algorithms, attempting to correct spelling errors and detect relevant words (e.g. a noun adds more meaning to a document than a determiner or preposition). Moreover, by using language technology we can relate morphological variants of words and compounds. For example, *huis* (English: *house*) and *huisje* (English: *small house*) represent the same concept and are both relevant terms for a document about houses, but have a different syntax. By relating such equivalent terms, we can improve the classification results. In conclusion we can state that language technology may assist in classifying e-mail, leading to our second research hypothesis:

2 The use of Language Technology (like stemming, part-of-speech tagging and spelling correction) as text normalization, improves the classification accuracy of the information retrieval based classification methods.

In this thesis we attempt to prove these two hypotheses by empirical research using an e-mail corpus acquired in a Dutch contact centre of "De National Postcodeloterij" (English: The national lottery of zip codes). This corpus is further described in section 8.1.1.

1.3 Main innovative aspects

The concept of using IR based classification techniques for text classification purposes is not new, neither is the concept of using text classification techniques to classify e-mail (for answer suggestion, or answer generation). However, using IR based classification techniques for e-mail classification is a new concept. Research on automatic classification of e-mail using text classification techniques has already been done in German and Dutch contact centres using Support Vector Machines (SVM, see section 4.2) and Neural Nets in German, and Naive Bayes (see section 4.1) in Dutch. This classification problem differs from text classification problems in the level of detail of the classes: text classification uses topics (like *sports* and *financial*) as classes, while this form of e-mail classification uses standard questions as classes. For the latter, two distinct questions may have the same topic, e.g. *How can I change my e-mail address* and *How can I apply for an e-mail address* are two distinct questions about the same topic (e-mail address).

The use of language technology to improve classification accuracy or information retrieval results (precision and recall) is not a new concept either. However, since this classification problem focusses on unstructured texts (e-mails) in particular, in which many spelling errors and grammatically inconsistencies may occur, it is not an equivalent problem to news article classification or abstract retrieval. The use of language technology improves accuracy on certain problems while it worsens the accuracy for others. Since e-mail classification (at this level) is not an equivalent problem to general text classification, and we use different classification techniques than the other e-mail classification research discussed in chapter 2, we may not assume that language technology yields the same results for our problem. Therefore we will also investigate the use of language technology for this specific problem.

1.4 Outline of the thesis

1. **Introduction** This chapter gives a brief introduction to the problem and states the two research hypotheses we attempt to prove in this thesis.

2. **Related work** In this chapter we will discuss several projects that are related to the research of this project.

3. Information Retrieval This chapter provides the theoretical basis for the information retrieval techniques used in this thesis.

4. Classification This chapter provides the theoretical basis for the classification techniques used in this thesis.

5. Natural Language Processing This chapter provides the theoretical basis for the language technologies that may improve classification accuracy.

6. Approach In this chapter we will present the chosen approach approach in detail. With this approach we attempt to prove the two hypotheses.

7 Design This chapter gives an overview of the algorithm used in our prototype and presents a detailed example to illustrate the working of these algorithms.

8. Evaluation This chapter we will discuss the experiments we performed.

9. Conclusions This chapter presents our conclusions with respect to the two research hypotheses.

10. Suggestions for future work Finally, we will point out some suggestions for further research on this topic.

Chapter 2

Related Work

In this chapter we will present some research that is (strongly) related to our e-mail answer suggestion problem. The first section covers projects that aim at classifying e-mail in contact centres: two for German and one for Dutch. The second section covers projects that are related because they deal with categorizing e-mail, albeit in a smaller and more general collection of folders or categories: two categories for spam filtering (spam and non-spam) and a handful of folders in an e-mail client. The last section covers more general text categorization problems, organised in the Text REtrieval Conference (TREC). Our e-mail answer suggestion problem is strongly related to such problems, since we use the analogy of text classification to suggest answers to e-mail.

2.1 E-mail answering assistance

Busemann, Schmeier, and Arens (2000) developed the ICC mail system to assist call centre agents in answering e-mails by suggesting relevant solutions for incoming e-mail. They use Shallow Text Processing (STP) like word stemming, part-of-speech tagging and sentence types, and Statistics-based Machine Learning (SML) techniques like lazy learners, neural networks and support vector machines for mapping incoming mail on standard answers. STP techniques are chosen above in-depth syntactic and semantic analysis, given the unstructured, informal and spontaneously created nature of e-mail, often containing misspellings, jargon and grammatical inaccuracy. Their experiments showed that the correct answer is selected in about 56.23% of the incoming e-mails (using support vector machines and shallow text processing). Neural networks and lazy learners only manage to select the correct text block (standard answer) in about 22% to 35% of the cases. Using support vector machines and STP, the correct text block is selected within the top 5 results in 78% of the cases.

Scheffer (2004) uses support vector machines (SVM) and Naive Bayes in a co-training learning environment to suggest possible answers for incoming e-mail in e-mail service centres. They map the problem to a semi-supervised text classification problem and automatically learn from in- and outbound e-mail. Bickel and Scheffer (2004) introduce the problem of automated e-mail answering in an e-mail service centre environment. This system learns from message pairs using a support vector machine, and tries to answer (instead of suggesting answers for) incoming e-mail based on previously given answers to similar questions. A clustering algorithm is used to cluster similar (training) e-mails and a SVM is trained for each cluster. Each cluster then contains all the training documents (e-mails) that represent one category. Incoming e-mail is compared (using the SVM) with each cluster (category), and the cluster with the highest similarity score (if above a certain threshold) represents the answer used to answer this e-mail. Their experiments show that the correct answer is selected in 40% of the e-mails. This percentage is increased to 50% when different answers for equal questions are merged (e.g. “When will my product be shipped” can be answered with “The product is already shipped” or “The product will be shipped tomorrow”), which require a different answer based on context information the customer does not know. Both answers rely on the same question, but differ based on the context information of when the product has been, or will be, shipped.

Gaustad and Bouma (2002) have experimented with an e-mail dataset acquired in a help desk environment in their research on Dutch text classification. Their dataset consisted of 6,000 e-mails, categorized in 193 categories, but their experiments focused on a subset of 5,518 e-mails categorized in 69 categories (covering 92.5% of the e-mails). For this dataset, the results ranged from 42.79% correct classification for the first suggestion of the system, to 77.65% correct classification in the best-5 results. For their experiments, Gaustad and Bouma use a Naive Bayes classifier, which is a simple but effective algorithm for text classification and e-mail classification (Androutsopoulos et al., 2000).

2.2 Spam filtering and e-mail categorization

Spam filtering can be treated as a text classification problem (with e-mail as documents) with two categories: spam and non-spam. Different classification approaches for spam filtering have been developed an experimented with, e.g. Naive Bayes (Androutsopoulos et al., 2000), Support Vector Machines (Drucker et al., 1999), memory based (IR like) methods using K-Nearest-Neighbour (Sakkis et al., 2003) and combinations of different approaches (Etzold, 2003). Many of these spam filtering techniques are used nowadays, to cope with the very large amount of spam sent daily.

E-mail categorization deals with routing incoming e-mail to user-defined folders. Most e-mail clients enable users to define rules for routing e-mail based on the sender, the topic and words in the content. Automatically classifying e-mail in folders has been an interesting research topic for years, and many different approaches have been tried (and proven successful). Boone (1998) has created an intelligent e-mail agent that can learn actions such as filtering, prioritising and forwarding e-mail using automatic feature extraction from text and meta-data like headers. Yang and Park (2002) have investigated the use of fast Machine Learning algorithms (like TF.IDF for the Rocchio classifier and Naive Bayes). Klimt and Yang (2004) introduce the Enron corpus as test bed for e-mail classification and perform initial baseline experiments using a support vector machine and features from the contents of e-mail as well as header information. Brutlag and Meek (2000) investigate the challenges in the email domain for text classification and compare SVM and IR classification (TF.IDF or Rocchio classification) while doing so. Their experiments show that little performance differences exist between these techniques for classifying e-mail in a set of user-defined folders.

2.3 Text categorization

The first Text REtrieval Conference (TREC) was held in November 1992 (Harman, 1993), with the intention to bring researchers together to discuss their work on a new large test collection. The TREC series have been brought to life with the goal of providing a large test collection for information retrieval (IR) purposes, that also reflects collections for large real-world information retrieval environments. This enables researchers to compare their findings with others based on the same test collection, instead of having to compare research that is performed on different collections. The first two tasks TREC addressed were fully aimed on information retrieval, and covered IR using an “ad hoc” query (e.g. in a library environment) and IR using a “routing” query (e.g. a filter to route some incoming information stream).

In later years, TREC also focussed on text categorization tasks, and developed several test sets for this purpose (based on the initial TREC test collection, in which documents are also provided with a topic). The Text REtrieval Conference series is a very useful resource for finding relevant information to text categorization and information retrieval problems. Many of the techniques discussed in this thesis have been presented at this conference and lots of research within this conference series focusses on text classification problems.

Chapter 3

Information Retrieval

Information retrieval (IR) has been a research topic since the early 1950's. The definition given by Mooers (1950) (recited from Hiemstra, 2000) is probably one of the most used definitions of information retrieval:

> *Information retrieval is the name of the process or method whereby a prospective user of information is able to convert his need for information into an actual list of citations to documents in storage containing information useful to him.*

Search engines on the internet are good examples of modern information retrieval systems that are widely used (e.g. Google and AltaVista). A user translates his need for information into a query and feeds it to the system (search engine). The search engine converts the query into a (ranked) list of relevant resources, in case of a search engine this will be web pages, but it could also be documents, pictures, etc.. The notion of relevance is very important in information retrieval. Unlike data retrieval (e.g. using a database query), where data, satisfying a specific set of constraints, has to retrieved, information retrieval aims at finding sources of information (not the information itself) that could satisfy the user's information need. Because many documents can be of some relevance to a users query, it is common to apply a ranking function to the results.

3.1 Information retrieval system

There are basically two types of information retrieval methods: Ad hoc and Filtering. If the document collection remains (relatively) static while new queries are submitted to the system, we speak of ad hoc retrieval. This is the most common form. An example of ad hoc information retrieval is the use of a search engine like Google, in which the user enters a search query to find relevant documents. A similar, but distinct, form is one in which the queries remain (relatively) static while new documents come into the system (the document collection is dynamic). This form of retrieval is called filtering, and is commonly based on a user profile describing the user's preferences, so new documents can be retrieved

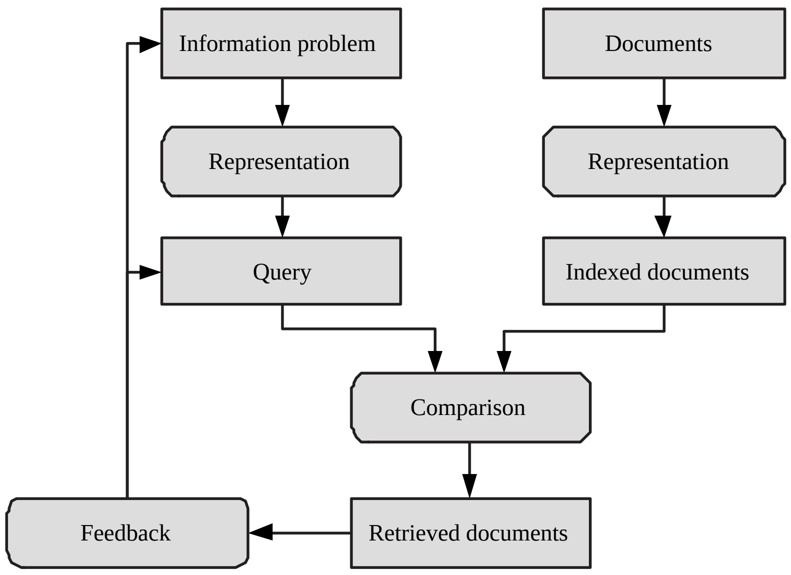

Figure 3.1: Basic processes in an IR system (Croft, 1993)

based on this profile (e.g. a news feed). For example, the user makes a profile of his or hers interest (e.g. soccer results of the English Premier League), and receives relevant information (e.g. news articles or websites) based on this profile.

An information retrieval system (a software program) consists of three basic processes: representing the content of the documents, representing the users information need and comparing both representations (Hiemstra, 2000). Figure 3.1 (recited from Hiemstra, 2000) visualizes these three processes. The squared boxes represent data and the rounded boxes represent processes.

The documents in the document collection are transformed into indexed documents by the representation process, also called the indexing process. Each document is transformed into a formal representation which the comparison process can understand, for example a vector representation (see section 3.2.3). These formal representations of the documents may be stored in a so called index (e.g. a database). Often, not the representations of the full documents are stored, but, for storage and efficiency reasons, only the title and abstract. In a more sophisticated system, all index terms may be weighted (see section 3.3), and these weights can also be stored in the index.

The user's information need is transformed into a query in the representation process, also called the query formulation process. Part of this process is performed by the user and part by the IR system. The user translates his information problem into a question, or set of keywords he or she wants to search on, called the initial query. The IR system translates this initial query to a formal representation that the system can understand. Based on the results of the comparison process, the query can be adjusted. This process of successive query formulation is called relevance feedback.

The main process of an IR system is comparing queries and documents. This is also called the matching process. In this process, the query representation is compared to all document representations in the index, resulting in a list of relevant documents. A document is marked relevant by the system solely based on the features represented in the document (words) and not by e.g. the writers intention. In ranked retrieval, the results (a list of documents) are ranked in decreasing order of relevance. Users can navigate this list in search of the information they need. Ranked retrieval intends to place the most relevant documents at the top of the list, minimizing the time the user has to invest searching for the desired documents.

3.2 Information retrieval models

Baeza-Yates and Ribeiro-Neto (1999) describe the three classic models in information retrieval: the Boolean model, the vector space model and the the probabilistic model. In the Boolean model, queries and documents are represented as sets of index terms (set theoretic model). In the vector space model queries and documents are represented as vectors in t-dimensional space (algebraic model). The probabilistic model presents a framework for representing queries and documents based on probability theory (probabilistic model). These classic models consider each document as a set of representative keywords called index terms. An *index term* is simply a word of a document whose semantics assists in remembering the documents main themes. Index terms can have weights assigned to them: Let k_i be an index term, d_j a document, and $w_{i,j} \geq 0$ be a weight associated with the pair (k_i, d_j). An index term that does not appear in the document is assigned a weight of 0.

More recently, statistical language models have been introduced by Hiemstra (2000) (amongst others). These models constitute a mathematical framework for the combination of natural language processing and information retrieval, and will be briefly discussed in section 3.2.5.

3.2.1 Formal characterization of an IR model

A formal characterization of an IR model is given by the next definition:

Definition An information retrieval model is a quadruple [**D**, **Q**, F, $R(q_i, d_j)$] where

(1) **D** *is a set composed of logical views (or representations) for the documents in the collection*

(2) **Q** *is a set composed of logical views (or representations) for the user information needs. Such representations are called queries*

(3) F *is a framework for modelling document representation, queries and their relationships*

(4) $R(q_i, d_j)$ *is a ranking function which associates a real number with a query* $q_i \epsilon$ **Q** *and a document* $d_j \epsilon$ **D**. *Such ranking defines an ordering among the documents with regard to the query* q_i.

3.2.2 Boolean model

The Boolean model is a simple retrieval model based on Boolean algebra. Drawbacks of the Boolean model are:

- Its retrieval strategy is based on a binary criterion: A document is predicted relevant or non-relevant without any notion of grading scale (it is much more a data retrieval model).
- It is difficult to translate information need into a Boolean expression, since Boolean expressions have very precise semantics.

The Boolean model considers that index terms are either present or absent in a document. As a result, the index term weights are assumed to be binary (0 or 1). A query q is a conventional Boolean expression: e.g. index terms linked by three connectives *not, and, or*. A document is judged either relevant or non-relevant by the Boolean model. A document is judged relevant, only if it matches the exact expression of the query.

Salton, Fox, and Wu (1983) have proposed and implemented an extended version of the original Boolean model in which ranked retrieval is also possible.

3.2.3 Vector model

According to Baeza-Yates and Ribeiro-Neto (1999), the vector space model admits that the use of binary weights is too limiting and proposes a framework in which partial matching is possible. The term weights are used to compute the degree of similarity between the user query and each document stored in the system. The index terms in the documents and query are weighted. Within the vector space model, documents and queries are represented as vectors in an *n*-dimensional space. Joachims (1997) explains a classification approach using the vector model and states that each document d_j is represented as a vector $\overrightarrow{d_j} = (d_j^1, ..., d_j^n)$. The query q is represented as a query vector $\overrightarrow{q} = (q^1, ..., q^n)$. Each element in a vector represents a distinct word w_i in the document collection. A term d^i can be assigned a binary weight (0 for absence and 1 for presence). This model also recognizes non-binary weights to distinguish between relevant and less-relevant words (see section 3.3). In this case a term d^i is represented by a positive weight equal or greater to 0 (a weight of zero represents the absence of a term). The higher the weight for a certain term, the more relevant this term is assumed to be.

The correlation (similarity) between the vectors representing the document and query, can be quantified by the cosine of the angle of the vectors, which is given in equation 3.1.

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|| \vec{d_j} || \times || \vec{q} ||} \tag{3.1}$$

Equation 3.1 divides the inner product of both vectors by the product of the Euclidean lengths[1] (also known as norm) of both vectors. The factor $||\vec{q}||$ does not affect the ranking of the results, since it is the same for all documents. The factor $||\vec{d_j}||$ provides a normalisation in the space of documents.

3.2.4 Probabilistic model

The classic probabilistic model was introduced in 1976 by Robertson and Sparck Jones (information taken from Baeza-Yates and Ribeiro-Neto (1999)) and later became known as the Binary Independence Retrieval (BIR) model. This model tries to capture the IR problem within a probabilistic framework. The idea is as follows: Given a user query, there is a set of documents which exactly contain the relevant documents and no other (ideal answer set). The probabilistic model is based on the Probabilistic Ranking Principle (PRP) (Robertson (1977)):

> If a reference retrieval systems response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.

Sparck Jones et al. (1998) argue that (following the PRP) for each document and each query the following basic question has to be answered:

> ”What is the probability that this document is relevant to this query?”

The probabilities for each query/document pair can then be ranked in order of their probability of relevance. In the probabilistic model, index term weight variables are all binary (0 or 1) for documents and queries. A query q is a subset of index terms. Let:

R	be the set of documents known (or initially guessed) to be relevant,
$\overline{R}$	be the complement of R (set of non-relevant documents),
$P(R, \vec{d_j})$	be the probability that the document d_j is relevant to the query and
$P(\overline{R}, \vec{d_j})$	be the probability that the document d_j is non-relevant to the query

[1]The Euclidean length of vector $\vec{d}$ with length N is $||\vec{d}|| = \sqrt{\sum_{i=1}^{N} d_i^2}$

then the similarity of a document and a query ($sim(d_j, q)$) is given by (using Bayes rule to simplify):

$$sim(d_j, q) = \frac{P(R|\vec{d_j})}{P(\overline{R}|\vec{d_j})} = \frac{P(\vec{d_j}|R) \cdot P(R)}{P(\vec{d_j}|\overline{R}) \cdot P(\overline{R})} \tag{3.2}$$

Where $P(R)$ stands for the probability that a document randomly selected from the document collection is relevant. $P(\vec{d_j}|R)$ stands for the probability of randomly selecting document d_j of the relevant documents. Since $P(R)$ and $P(\overline{R})$ are constant for every document, and therefore do not influence the ranking of documents, we can write:

$$sim(d_j, q) \sim \frac{P(\vec{d_j}|R)}{P(\vec{d_j}|\overline{R})} \tag{3.3}$$

In the probabilistic model it is assumed that index terms are independent. Therefore we can rewrite the probability that a document containing n index terms is relevant as the product of the probabilities that each of the independent index terms (k_i) is relevant using equation 3.4:

$$P(\vec{d_j}|R) = \prod_{i=1}^{n} P(k_i|R) \tag{3.4}$$

Combining 3.3 and 3.4 results in equation 3.5:

$$sim(d_j, q) \sim \frac{\prod_{g_i(\vec{d_j})=1} P(k_i|R)) \cdot \prod_{g_i(\vec{d_j})=0} P(\overline{k}_i|R))}{\prod_{g_i(\vec{d_j})=1} P(k_i|\overline{R})) \cdot \prod_{g_i(\vec{d_j})=0} P(\overline{k}_i|\overline{R}))} \tag{3.5}$$

Where,

$P(k_i\|R)$	stands for the probability that the index term k_i is present in a document randomly selected from R,
$P(\overline{k}_i\|R)$	stands for the probability that the index term k_i is not present in a document randomly selected from R, and
$g_i(\vec{d_j}) = 1$	means that the weight assigned to term j from document d equals 1 (a similar explanation can be given if the weight equals 0).

This may seem a rather complicated function, but intuitively it is not. It just states that the similarity between a document and a query depends on the index terms in the query and a document. If an index term has been assigned a weight of 0, it is not present in the document and if an index term has been assigned a weight of 1, it is present in the document. This equation multiplies the probabilities that index terms with weight 1 (present in the document) are relevant with the probabilities that index terms with weight 0 (not present in the document) are relevant. A similar explanation can be given for the denominator with the small distinction that probabilities on non-relevance are calculated. This equation can be somewhat simplified by recalling that $P(k_i|R) + P(\overline{k_i}|R) = 1$, taking logarithms and ignoring constant factors (for all documents in the context of the same query), resulting in the next ranking expression in the probabilistic model:

$$sim(d_j, q) \sim \sum_{i=1}^{n} w_{i,q} \cdot w_{i,j} \cdot (log \frac{P(k_i|R)}{1 - P(k_i|R)} + log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})}) \tag{3.6}$$

Where $w_{i,q}$ and $w_{i,j}$ represent the weights of index terms of the query and document respectively.

3.2.5 Statistical language models

Hiemstra and de Jong (2001) state that full-text information retrieval is all about natural language understanding. Common information retrieval systems only use word statistics, and hence treat words like *cow*, *cows*, *cattle* and *milk* as if they were totally unrelated. Information retrieval systems might benefit from such relational information (*cows* is the plural of *cow* and *cattle* is the hypernym of *cows*) by retrieving relevant documents containing terms that are not literally present in the query. Common information retrieval systems use simple language technology like stemming and stop word removal, but Hiemstra and de Jong claim that many of the near future information retrieval applications require for more serious natural language processing. For this reason they presented a mathematical framework for statistical language models in information retrieval.

The concept of these language models is assigning probabilities to sequences of words. If a word occurs three times in a document of 100 words, the probability of that word, given that document is 0.03, and if a word occurs 2000 times in a corpus of a million words, then the probability of that word is 0.002. For IR, a language model is defined for each separate document in the collection, modelling the typical language use on that particular document. $P(D)$ is the probability of the event that a document is relevant, where D can be any document in the collection. $P(T)$ is the probability that the term T occurs in the relevant document's language. It is assumed that some words of a user query are important, while others are not. Since we cannot know beforehand which are the important words (in general or from the document in specific), the model uses a mixture of the probability $P(T)$ (e.g. a term is relevant for a given collection of documents) and the probability $P(T|D)$ (a term in the relevant document). The basic model assumes independence between query terms in a sequence of n terms $T_1, T_2, ..., T_n$, resulting in the definition of the language model given in equation 3.7.

$$P(T_1, ..., T_n|D) = \prod_{i=1}^{n} ((1 - \lambda_i)P(T_i) + \lambda_i P(T_i|D)) \tag{3.7}$$

The parameter λ_i $(0 \leq \lambda_i \leq 1)$ defines the probability that a term is important. In an IR application documents are ranked in decreasing order of this probability.

Hiemstra (2000) shows that this model outperforms ranked retrieval models that use today's best performing TF.IDF weighting variations (see 3.3).

Besides its good performance, this model enables us to model the use of language technology in information retrieval systems. Stopword removal (see section 5.1) can be modelled by setting λ_i to 0, so that it does not influence the ranking results. If $\lambda_i = 0$, the term T_i contributes the same amount of probability to the final result of the computation, whether it occurs in the document or not. Another language technology that might be modelled is the use of simple phrases instead of single terms, using N-gram models. For instance, the sentence *stock exchange* carries more information than just the single words *stock* and *exchange*, which may easily lead to documents about the exchange of live stock, instead of documents about Wall Street. Simple sentences of 2 words may be modelled using the bi-gram model: instead of using the probability $P(T_i)$ we now use the probability $P(T_i|T_{i-1})$, which means that the probability that term T_i (*exchange*) occurs, depends on the occurrence of the preceding term T_{i-1} (*stock*).

3.3 Term weighting

Not all terms in a document are equally important in an IR system. Some words (like *the*, *or* and *besides*) are not nearly as relevant as other (more specific) words like for instance *bicycle* or *computer*. To distinguish between relevant and non-relevant words, term relevance weighting schemes have been suggested to improve the results of IR systems. By using a term relevance weighting scheme we can specify which terms are important and which are less important.

3.3.1 TF.IDF weighting

In order to find relevant documents to a query, we first have to know what makes a document relevant. Recall that a document is represented by a set of index terms, which determines the relevance of the document. We can now define two characteristics that determine the relevance of a term:

- A term is important if it occurs a lot in one document
- A term is distinctive if it occurs in as few documents as possible

The first characteristic is obvious: if a document contains a certain word many times (e.g. *bicycle*), the document will probably contain enough information about that word (e.g. bicycles), making the word an important term. The second characteristic will be explained by example. Suppose a user searches for information about bicycles, and formulates the query: *information on new bicycles*. The words *new* and *on* are words that occur in many documents, and in many contexts, but do not help on finding information about a specific topic (e.g. bicycles). These words simply do not have enough distinctive power compared to a word like *bicycle*, that occurs mostly in documents about bicycles.

This ideology led to the famous TF.IDF weighting scheme. Salton and McGill (1983) describe that these important issues for relevance weighting can be described by *term frequency* and *inverse document frequency*. The term frequency $TF(w_i, d)$ is the number of times word w_i occurs in document d. The document frequency $DF(w_i)$ is the number of documents in which word w_i occurs at least once. The inverse document frequency $IDF(w_i)$ determines the specificity of a term:

$$IDF(w_i) = \log\left(\frac{|D|}{DF(w_i)}\right) \tag{3.8}$$

Where $|D|$ is the total number of documents in the collection. Intuitively, the IDF is low if a term occurs in many documents, and is highest if it occurs in only one. Each term $d^{(i)}$ in a document d_j can now be weighted using the TF.IDF weighting scheme according to equation (3.9):

$$d_j{}^{(i)} = TF(w_i, d_j) \times IDF(w_i) \tag{3.9}$$

Determining the similarity between a document and query vector can also be done using weighted vectors in the cosine similarity measure, as in equation 3.10, using the definitions of the inner product and Euclidean length.

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{\|\vec{d_j}\| \times \|\vec{q}\|} = \frac{\sum_{i=1}^{n} d_j{}^i \times q^i}{\sqrt{\sum_{i=1}^{n} (d_j{}^i)^2} \times \sqrt{\sum_{i=1}^{n} (q^i)^2}} \tag{3.10}$$

3.3.2 Okapi weighting

The Okapi system was developed in the 1980's at the Polytechnic of Central London and later developed at City University London and Microsoft Research. The system is based on the probabilistic information retrieval model. Robertson and Walker experimented with additional weighting algorithms for this system, leading to the BM25 formula (BM stands for Best Match). Hiemstra (2000) describes that the Okapi weighting algorithm uses weights that are approximately linear for small values of term frequency tf, but do not increase in the same rate for larger values of tf.

Relevance weighting

In this system, documents and queries are also represented by vectors and their similarity is determined by the vector product (inner product) in equation (3.11):

$$score(\vec{d}, \vec{q}) = \sum_{k=1}^{m} d_k \bullet q_k \tag{3.11}$$

Robertson and Sparck Jones (1997) give a simple and clear explanation of the Okapi weighting algorithm (that is given below) in their technical report. Within this algorithm there are three different sources of weighting data. The first one is called *collection frequency weight* of an index term w_i (like the document frequency in TF.IDF weighting) and is given by:

$$CFW(w_i) = \log N - \log n \tag{3.12}$$

Where n is the number of documents a term d_j occurs in and N the total number of documents in the collection. The second source of weighting data is the *term frequency*: $TF(w_i, d)$ which just like in TF.IDF weighting equals the number of occurrences of term w_i in document d.

The third source of weighting data is the *document length* which, in the Okapi system, must be related to the term frequency of a term in a document. The motivation behind this weighting source is that if a term occurs in a short document and in a long one for the same number of times, the occurrence in the short document is likely to be more valuable. The document length $DL(d)$ is the total number of term occurrences in a document d. The document length can be normalized by dividing the the document length by the average document length of all documents (ADL) and has the advantage that the units in which the document length is counted does not matter too much (e.g. the total number of characters can also be used):

$$NDL(d) = \frac{DL(d)}{ADL} \tag{3.13}$$

These three sources of weighting data can be combined into a single formula giving the *combined weight* for a term w_i by equation (3.14):

$$CW(w_i, d) = \frac{CFW(w_i) \cdot TF(w_i, d) \cdot (k_1 + 1)}{k_1 \cdot ((1-b) + (b \cdot (NDL(d)))) + TF(w_i, d)} \tag{3.14}$$

This combined weight indicates the relevance of each term for a document. In the above equation, k_1 and b are tuning constants. Tuning constant k_1 modifies the extent of the influence of term frequency where b (which ranges from 0 to 1) modifies the effect of documentg length. $b = 0$ assumes that (long) documents are long because they are multi-topic (e.g. cover several topics) and $b = 1$ assumes that documents are long because they are repetitive (cover the same topic over again). Robertson and Sparck Jones (1997) state that $k_1 = 2$ and $b = 0.75$ are good initial values for this weighting scheme, but experiments should be done to determine optimum values for specific applications.

Query expansion and iterative searching

Robertson and Sparck Jones (1997) also describe some methods of query expansion. Based on the initial results of the system, the query can be expanded with search terms that are assumed relevant. Besides expanding the query, the combined (relevance) weights $(CW(w_i))$ are recalculated based on the results by the next formula:

$$RW(w_i, d) = \log \frac{(r+0.5)(N-n-R+r+0.5)}{(n-r+0.5)(R-r+0.5)} \tag{3.15}$$

Where r is the number of known relevant documents containing term w_i and R is the total number of known relevant documents to a query. This formula can be used instead of $CFW(w_i)$ for all terms used in a second or subsequent iteration. The process of refining the query based on the results of the previous iteration is called *iterative searching*.

Longer queries

The weighting algorithm described has good performance for queries with at least 5 words. However, for longer queries Robertson and Sparck Jones (1997) have also experimented assigning weights to query terms. If a query is longer than a sentence (or a few words) the Query Adjusted Combined Weight should be computed according to equation (3.16):

$$QACW(w_i, d) = QF(w_i) \cdot CW(w_i, d) \tag{3.16}$$

Where $QF(w_i)$ is the number of occurrences of term w_i in the query.

3.4 Evaluation metrics

Van Rijsbergen (1979) already stated that the evaluation of an information retrieval system is related to the relevance of the documents in the collection and the retrieved documents by an IR system. A document can either be retrieved or not, and be relevant or not. Let us define the relevant set of documents in a collection as A and the non-relevant set of documents as $\overline{A}$. In a similar way we define the set of retrieved documents in a collection by an IR system as B and the set of non-retrieved documents by $\overline{B}$. Table 3.1 displays the 'contingency' table for these measures. N represents the total set of documents in the collection.

	RELEVANT	NON-RELEVANT	
RETRIEVED	$A \cap B$	$\overline{A} \cap B$	B
NOT RETRIEVED	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$	$\overline{B}$
	A	$\overline{A}$	N

Table 3.1: Recall and precision contingency table (Van Rijsbergen, 1979)

Three evaluation measures can be defined based on the data in table 3.1. *Precision* (equation 3.17) is the fraction of retrieved documents which is relevant.

$$Precision = \frac{|A \cap B|}{|B|} \tag{3.17}$$

Recall (equation 3.18) is the fraction of the relevant documents which has been retrieved.

$$Recall = \frac{|A \cap B|}{|A|} \tag{3.18}$$

Fallout (equation 3.19) is the fraction of the non-relevant documents which has been retrieved

$$Fallout = \frac{|\overline{A} \cap B|}{|\overline{A}|} \tag{3.19}$$

Recall and precision are common used evaluation metrics for information retrieval systems. The average precision at different recall levels (e.g. at 0%, 10%, ...,100%) is called the 11pt precision. In an IR system a good trade off has to be found between the recall and precision metrics. If al documents in a collection are returned on a query, recall is 100%, but precision is dramatically low, and vice versa if none of the documents is returned.

Chapter 4

Classification

In chapter 1 we assumed that within the e-mail answer suggestion problem, similar questions require a similar answer. If we cast the e-mail answer suggestion problem to a text classification problem, this assumption induces that similar questions should be categorized in the same category (which represents a standard question). In text classification, the problem is to assign a predefined category to each document in the collection, based on the example of (manually) pre-categorized documents. Like in information retrieval, we use words as the features of a document (we represent them in this chapter as $< f_1, ..., f_n >$). In this chapter we will discuss three approaches to text classification.

4.1 Naive Bayes

Naive Bayes is a rather simple yet effective statistical classification method which has been widely used for text classification (Gaustad and Bouma, 2002). Naive Bayes estimates the probability that a given document, containing features $f_1, ..., f_n$ belongs to a class $c \in C$ using equation 4.1.

$$P(c|f_1, ..., f_n) = \frac{P(c) \cdot \prod_{i=1}^{n} P(f_i|c)}{\sum_{k \in C} P(k) \cdot \prod_{i=1}^{n} P(f_i|k)} \tag{4.1}$$

The Naive Bayes approach is naive in the sense that it assumes that words (features) are independent of one another. This assumption usually is not true, since words are not completely independent of one another, but estimating the probability that a certain document (represented by a feature vector $< f_1, ..., f_n >$) should be classified in a certain category, taking into account all the dependencies between these words is simply not feasible. Using the independence assumption and Bayes Law[1] we can estimate the probability $P(f_i|c)$ relatively straightforward (by counting the number of times feature f_i occurs in class c).

[1] $P(c|x) = \frac{P(c)P(c|x)}{P(x)}$, but since the denominator is constant for all classes, we omit it in equation 4.1

4.2 Support vector machines

Support Vector Machines (SVM's) are first introduced by Cortes and Vapnik (1995) and are very suitable for text classification methods (Joachims, 1998). Joachims treats a text classification problem as a set of separate binary text classification problems (one for each category). As document representation he uses words as features, but only if these words occur at least three times and are not stopwords (like *and*, *of* and *the*), and scales these feature vectors using inverse document frequency (see section 3.3).

Support vector machines are based on the *Structural Risk Minimization* principle, which tries to find a hypothesis h which can guarantee the lowest true error. The true error of h is the probability that h will make an error on an unseen randomly selected test example (i.e. a random document that should be classified). SVM's basically learn linear threshold functions, but can easily be used to learn more difficult functions like polynomial ones. Typically, a basic SVM may learn a binary distinction function, visualized in figure 4.1. The support vector machine relies on the support vectors to separate the data into two groups: one that belongs to class 1, and one that does not. The support vector machine chooses it margins as large as possible, and tries to find the Optimum Separating Hyperplane (OSH), denoted by the support vectors (using the dashed lines in the figure), without creating classification errors. Notice that this figure represents the best case: the data can be separated with no errors (e.g. a dot is not separated from the crosses, but from the other dots).

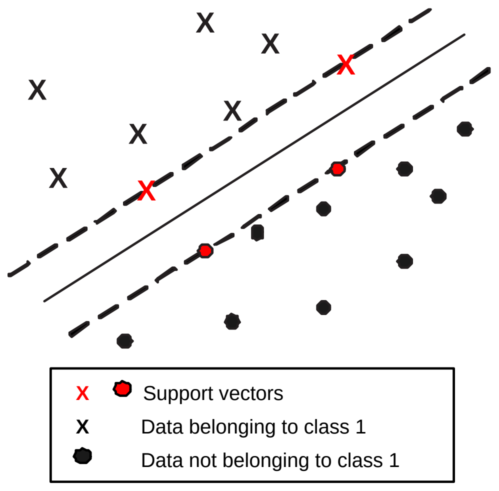

Figure 4.1: Two-dimensional linear threshold function that can be solved using SVM's

A remarkable property of SVM's is that their ability to learn can be independent of the dimensionality of the feature space, since they base the complexity of the hypotheses on the margin they separate the data and not the number of features. This means that the best set of features are used to separate the data, and not all features (see figure 4.1).

SVM's should work well for text categorization because of this independence of the dimensionality of the feature space and text classifiers have to deal with feature spaces of over 10000 features. Moreover, most text categorization problems are linearly separable, and SVM's try to find such linear separators.

Joachims experimented with SVM's on the Reuters-21578 dataset (see section 6.2) and found that SVM's outperformed Naive Bayes (section 4.1), Rocchio (section 4.3.1) and K-NN classification (section 4.3.2). Joachims also adjusted his Support Vector Machines so that they can output a ranked list of results, of which the most suitable category is placed at the top (Joachims, 1999).

4.3 IR based classification

Sebastiani (1999) describes two distinct approaches for information retrieval based classification. The first approach clusters all documents belonging to one category into a single representation and compares these category representations to the query. The category representation with the highest relevance score is most likely the category to which the query belongs. Such a classifier is called a *profile-based classifier*. The second approach compares the query to every document in the document collection and determines the category that is most likely correct for this query, based on the most relevant documents. E.g. category 1 occurs seven times in the top ten results for a query (new document), so category 1 is most likely the correct category for our new document. Such a classifier is called an *example-based classifier*.

4.3.1 Profile based classification

A profile-based classifier is basically a classifier which embodies an explicit, or declarative, representation of the category on which it needs to take decisions. Rocchio's classifier is the foremost example of such a classifier. Rocchio developed an algorithm for relevance feedback for use in the vector space information retrieval model, which can be adapted to serve as a profile-based classifier. Joachims (1997) describes the use of the Rocchio classifier using TF.IDF weights, but other weighting schemes may also be used. Recall from section 3.3.1 that documents are represented by a vector $\vec{d_j}$ containing the weights of index terms (calculated using the TF.IDF weighting scheme). First, the classifier should learn to classify documents. This is achieved by combining document vectors (of one category) into a prototype vector $\vec{c_j}$ for each class C_j. In this training phase, both the normalized vectors of the positive examples for a class as well as those of the negative examples of a class are used. The prototype vector is then calculated as a weighted difference of the positive and negative examples (as can be seen in equation 4.2).

$$\vec{c}_j = \alpha \left(\frac{1}{|C_j|} \sum_{\vec{d}_j \in C_j} \frac{\vec{d_j}}{\| \vec{d_j} \|} \right) - \beta \left(\frac{1}{|D - C_j|} \sum_{\vec{d}_j \in D - C_j} \frac{\vec{d_j}}{\| \vec{d_j} \|} \right) \tag{4.2}$$

Where, α and β are parameters that adjust the relative impact of positive and negative training examples (recommended to be 16 and 4 respectively). Furthermore, C_j is the set of training documents assigned to class j and $||\overrightarrow{d_j}||$ denotes the Euclidean length of a vector $\overrightarrow{d_j}$. Additionally, Rocchio requires that negative elements of the vector c_j are reset to 0. The resulting set of prototype vectors (one for each class) represents the learned model that can be used to classify a new document d' using equation (4.3).

$$H_{TFIDF}(d') = \underset{C_j \in C}{\arg\max}\; cos(\overrightarrow{c_j}, \overrightarrow{d'}) \tag{4.3}$$

The classification function H_{TFIDF} (H for hypothesis) returns the category that has the highest similarity score (using the cosine function, but other similarity functions may also be used) with respect to the document to be classified. This approach can be slightly adjusted to return a ranked list (in decreasing order of similarity) of categories that are suitable for document $\overrightarrow{d_j}$ by ignoring the $\arg\max$ function and ordering the calculated similarity scores for each category in decreasing order (cut off at a certain threshold if pleased).

4.3.2 Example based classification

Example based classifiers do not build a representation for each category but use the categorization judgements that experts have given on the training documents similar to the one to be categorized. Such classifiers are therefore called lazy learning systems, since they do not involve a true training phase. A commonly used algorithm for example-based classification is the K-NN (K-Nearest-Neighbour) algorithm, implemented by Yang (1994) in the Expert System. The conditional probability that a document d_j is classified in category c_k by human judgement, is given by equation 4.4:

$$Pr(c_k|d_j) \approx \frac{\#(assign(c_k, d_j))}{\#(d_j \in D)} \tag{4.4}$$

Where $d_1, ..., d_m$ are unique training documents and $C_1, .., C_l$ are unique categories. Furthermore, $\#(assign(c_k, d_j))$ is the number of times category c_k is assigned to document d_j and $\#(d_j \in D)$ is the number of times document d_j occurs in the document collection D. This probability is calculated since a document may have more than one occurrence in the training sample (at least after text normalization like stopword removal and stemming). Usually this equation results in a 0 or 1, indicating a category is or is not assigned to a document. The relevance score is then calculated (using equation 4.5) by comparing the query q to all documents $d_j \in D$ using a similarity measure like the inner product or cosine, and multiplying the result with the conditional probability calculated before.

$$rel(c_k|q) \approx \sum_{j=1}^{m} sim(q, d_j) \times Pr(c_k|d_j) \tag{4.5}$$

Where $sim(q|d_j)$ is the similarity score calculated by the IR component and both $sim(q|d_j)$ and $rel(c_k|q)$ are scores, not probabilities. If $Pr(c_k|d_j)$ is either 0 or 1, this formula just adds the similarity scores calculated for the document to be classified and each document in the document collection. To use this classification formula for K-NN classification, we can adjust equation 4.5, by summing the results of only the top K ($1 \leq K \leq m$) documents retrieved by the IR component, to equation 4.6:

$$rel(c_k|q) \approx \sum_{j=0}^{K} sim(q, d_j) \times Pr(c_k|d_j) \tag{4.6}$$

The results of calculating the relevance of a category to a given document (as in equation 4.6) can be used to return the most relevant document as the category the new document has to be categorized in, or return a ranking (in descending order of relevance) of categories most suitable for the new document. This ranking can be cut off at a certain threshold (based on the application and user's need).

4.4 Evaluation metrics

Yang and Liu (1999) use the F_1 measure (initially introduced by Van Rijsbergen (1979) which combines recall (r) and precision (p) with an equal weight in the following form:

$$F_1(r, p) = \frac{2rp}{r + p} \tag{4.7}$$

These scores can be computed for the binary decisions on each individual category first and then be averaged over categories. Or, they can be computed globally over all the $n \times m$ binary decisions where n is the number of total test documents, and m is the number of categories in consideration.

Another common measure for classification problems is the classification accuracy, which is simply given by the percentage of documents that is correctly classified. E.g. if 60 of the 100 documents are categorized in the correct category, the classification accuracy (or simply performance) is 60%. In section 8.1.2 we give a definition of performance for ranked classification systems (in which the best-n categories are given in a descending order of relevance)

Chapter 5

Natural Language Processing

In our second research hypothesis we stated that the use of natural language processing (NLP, or language technology) which could be used as a text normalization technique, improves classification accuracy in the e-mail classification problem. In this chapter we describe seven techniques of NLP that might help realising this.

5.1 Stopword removal

Zipf's law states that the product of the frequency of use of words and the rank order (if we rank these words in descending order of frequency) is approximately constant (Van Rijsbergen, 1979), meaning a large part of a text consists of very few words. Most of these very frequent words are stopwords. A stopword is a word which does not carry meaning in a natural language (Baeza-Yates and Ribeiro-Neto, 1999). Typical candidates for stopwords are articles, prepositions and conjunctions. For instance, in English: *of*, *the*, *and*, *a*, *to*, *in* and in Dutch: *en*, *de*, *een*, *omdat*, *desalniettemin* are typical stopwords.

Eliminating these words should not significantly influence the classification accuracy or retrieval precision, because these words do not carry meaning. Moreover, words occurring in a great amount of all documents, are fairly useless in information retreival (see chapter 3). However, if we remove stopwords, we typically obtain a compression in the size of the indexing structure of 40% (e.g. in the inverted file index in section 7.3).

Stopword removal can simply be implemented by using a stopword list. If a word occurs in this list, we remove it from the document (or simply do not use it in indexing and classification). Despite the benefits, stopword removal might reduce recall in IR systems (and accuracy in classification systems), for instance if the search query for Shakespeare's work is denoted by only the famous quote *to be or not to be*. Eliminating stopwords from that query only leaves the word *be*, making it very difficult to find relevant documents to this query.

5.2 Stemming

Kraaij and Pohlmann (1996) have investigated the effect of stemming of document and query terms in information retrieval systems. Stemming is the process of mapping different morphological variants (word forms) to a single stem (e.g. *swimming* and *swimmer* can both be reduced to their stem *swim*). Kraaij and Pohlmann state that by reducing the morphological variance of terms (e.g. mapping single and plural forms of the same word on a single stem), researchers hope to improve the query-document matching process. They specifically investigated the effectiveness of suffix striping for the Dutch language and what effect the use of more linguistically motivated stemming techniques would have.

5.2.1 Suffix striping and the Dutch Porter stemmer

Suffix striping is one of the simplest stemming techniques. It uses a list of frequent suffixes and a set of rules to reduce words to their stem. Examples of a such a stemmer are the Lovins Stemmer (Lovins, 1968) and the famous Porter stemmer (Porter, 1980). For English, the use of stemming is somewhat controversial: according to Harman (1991), who investigated the Lovins stemmer, Porter stemmer and the S-stemmer[1], none of these algorithms consistently improved performance. In the morphologically more complex language Slovene, Popovic and Willet (1992) found more favourable results of Porter-like stemming. However, translating the Slovene corpus to English, and applying stemming to the translated corpus did not improve retrieval, confirming Harmans conclusion. Finally, Hull (1996) shows that a more detailed evaluation, focussed on recall, does reveal significant improvement, even for English.

Kraaij and Pohlmann (1994) have developed a stemmer for the Dutch language based on Porter's stemming algorithm. Porter's algorithm is based on a series of steps that each remove a certain type of suffix by way of substitution rules. These rules only apply when certain conditions hold, e.g. the resulting stem must have a certain minimum length. The original Porter stemmer only treats suffixes, but the Dutch Porter stemmer also deals with the pre- and infixes *ge*, which is introduced in most Dutch past particles (e.g. in *gezwommen*, English: *swum*). The use of the Dutch Porter stemmer in an information retrieval system increases recall, but at the cost of precision.

5.2.2 Dictionary based stemming

A Porter-like stemming algorithm is easy to implement and less time-consuming in processing the documents, but also brings along a lower stemming accuracy. Nevertheless, this type of stemming is usually sufficient for information retrieval systems. To increase stemming accuracy, a dictionary based stemmer may be used, either with or without a stemming algorithm as backup (in case no dictionary entry is found).

[1]a simple stemmer conflating single and plural word forms

Gaustad and Bouma (2002) have developed a dictionary based stemmer with a rule-based backup that outperforms the Dutch Porter stemmer in terms of accuracy, without being substantially slower. The dictionary based stemmer they use is based on the Celex database (Baayen et al., 1993) and yields a stemming accuracy of 96.27%, while the Dutch Porter stemmer yields an accuracy of 79.23%. In their experiments on e-mail classification, both dictionary based and Dutch Porter stemming do not significantly improve or worsen the classification results. Experiments on a Dutch news articles corpus taken from *De Volkskrant* showed that stemming did not have clear effect on classification accuracy, and differences between the Dutch Porter stemmer and the Dictionary based stemmer remain small.

5.3 Spelling correction

Spelling suggestion and spelling correction are common used tools in word processing programs, and can also be used to correct spelling errors in incoming e-mail in the contact centre. If a misspelled word occurs in an e-mail, determining if mails are similar becomes more difficult since the spelling of the words differs (even though the same word is meant). Jurafsky and Martin (2000) describe several techniques to detect and correct spelling errors, which will be discussed in this section. Of all misspelled words, 80% are caused by single-error misspellings, in particularly one of errors below:

insertion:	mistyping *the* as *ther*
deletion:	mistyping *the* as *th*
substitution:	mistyping *the* as *thw*
transposition:	mistyping *the* as *hte*

5.3.1 Probabilistic models of spelling correction

Probabilistic models of spelling correction are based on the Noisy Channel model (Jurafsky and Martin, 2000). The input of this model is the correct spelling of a word, which is transformed into a misspelled word by transportation through the noisy channel (e.g. a human being by mistyping a character). Such a channel introduces “noise” which makes it hard to recognize the correct (and intended) word. The goal of probabilistic models for spelling correction is to figure out the correct sequence of letters for a word, based on the “noisy” representation of that word. The most used probabilistic model for such problems is based on Bayesian Inference, which can also be used for classification problems (as a Naive Bayes classifier, see section 4.1).

As an example we use the approach of Kernighan, Church, and Gale (1990), who assume that the correct word will differ from the misspelling by a single insertion, deletion, substitution or transposition (which should cover most of the misspellings). First of all, we need to detect words that have most likely been misspelled, for instance by searching a dictionary (notice that this method only works for non-word spelling errors, i.e. where the mistyped word is not a real word any more like *hte* for *the*). As an example of all transformations they use

Typo	**Correction**	**Transformation**			
acress	actress	@	t	2	deletion
acress	cress	a	#	0	insertion
acress	caress	ac	ca	0	transposition
acress	access	r	c	2	substitution
acress	across	e	o	3	substitution
acress	acres	s	#	4	deletion

Table 5.1: Example of possible corrections of the word *acress* (Kernighan et al., 1990)

the word *acress* in table 5.1, for instance, the correct word *actress* could be transformed by replacing the *t* with nothing (@) at position 2[2].

The second stage of the algorithm scores each correction by use of equation 5.1. Let *t* be the typo, and let *c* range over the set *C* of candidate corrections. The most likely correction is then given by equation 5.1[3].

$$\widehat{c} = \arg\max_{c \in C} P(t|c)P(c) \tag{5.1}$$

The statistical data for estimating $P(c)$ is collected by counting the number of occurrences of word c in a corpus, but he statistical data for estimating $P(t|c)$ (likelyhood of the typo) is more difficult to estimate. This estimation is based on the number of times that character *e* was substituted for character *o* in large corpora, providing an estimate for $P(acress|across)$.

If real word errors (the intended word is transformed in another correct word, like *dessert* into *desert*) should also be corrected, context-sensitive spelling error corrections should be used (for more information we refer to Jurafsky and Martin, 2000). It is estimated that 15% of the single typographical errors produce valid English words.

5.3.2 Levenshtein Distance

Using the Levenshtein Distance, we can also detect and correct misspelled words of which more than one letter is misspelled (unlike Kernighans approach). The Levenshtein distance between two words is the minimum number of editing operations (insertion, deletion and substitution) needed to transform one word into the other (Jurafsky and Martin, 2000). Each operation is assigned a weight (usually 1) For example, the distance between *intention* and *execution*, using a weight of 1 for all operations, is 5:

[2] The symbols @ and # represent nulls in the typo and correction respectively. The transformations are named from the point of view of the correction, not the type

[3] Bayes equation normally has a denominator: $\arg\max_{c \in C} \frac{P(t|c)P(c)}{P(c)}$, but since the denominator is constant for all $c \in C$, it is omitted

		`intention`
1	delete *i*	`tention`
2	substitute *n* by *e*	`etention`
3	substitute *t* by *x*	`exention`
4	insert *u*	`exenution`
5	substitute *n* by *c*	`execution`

The best suggestion for the misspelled word is that of which the Levenshtein distance is lowest. For more complex Levenshtein models, we can also use probabilities that estimate the cost for each operation (similar to Bayes inference for spelling correction).

5.4 Language identification

Language identification is the process of determining the language of document, a sentence or even a word. Cavnar and Trenkle (1994) use N-grams for language identification, which is a process of text classification in which documents should be categorized in languages. They define an N-gram as an N-character slice of a longer string (for contiguous slices). For instance, the word text is composed of the following N-grams (in which we use the underscore character to represent blanks):

bi-grams (N = 2)	_T , TE, EX, XT, T_
tri-grams (N = 3)	_TE, TEX, EXT, XT_ , T_ _
quad-grams (N = 4)	_TEX, TEXT, EXT_ , XT_ _ , T_ _ _

Cavnar and Trenkle use a corpus for each different language, in which they count the occurrences of all possible character N-grams. This statistical information is needed to calculate the probability that a document (or word or sentence) is written in a certain language, by using Markov Chains: An N-gram is actually an N-th order Markov Chain (Jurafsky and Martin, 2000).

In order to estimate the probability that a certain word belongs to a certain language without using very large dictionaries for each language, we can use character sequences that are specific to languages. Therefore, we take a corpus for each language, and collect statistical information of all possible character sequences, from which we can estimate the probability that a word belongs to the specific language by using equation 5.2 (c_1^n represents a character sequence of n characters and 5.2 is simplified using the chain rule).

$$\begin{aligned} P(c_1^n) = P(c_1, c_2, ..., c_{n-1}, c_n) = \\ P(c_1)P(c_2|c_1)P(c_3|c_1^2)...P(c_n|c_1^{n-1}) \end{aligned} \tag{5.2}$$

Since we are not able to create corpora large enough to store all possible character sequences for a certain language, we estimate the probability following the assumption that the probability that a character occurs, is only based on the preceding *N* characters. Such an approximation model is called an N-th order Markov Chain and is modelled by equation 5.3

$$P(c_1^n) = \prod_{k=1}^{N} P(c_k|c_1^{k-1}) \tag{5.3}$$

If we assume that the occurrence of a character is only dependent of the preceding two characters (for which we model N-grams of a length of 3) we can model the probability that a certain character occurs using the 3-th order Markov Chain from equation 5.4.

$$P(c_1^n) = \prod_{k=1}^{N=3} P(c_k|c_1^{k-1}) = P(c_1)P(c_2|c_1)P(c_3|c_2, c_1) \tag{5.4}$$

N-th order Markov Models can be used in information retrieval and text classification to estimate the probability that a word is from a certain language. If this probability does not exceed a certain threshold (e.g. 0, which indicates that a certain character sequence does not occur in that language corpus), we could opt not to index this word, neglecting it in the retrieval or classification process (just like stop words).

5.5 Decompounding

Dutch (like German, Finnish and Swedish) is a compounding language, meaning words may be formed by concatenating other words in a productive process. For instance a board meeting is called a *directievergadering* in Dutch, which is a concatenation of *directie* (English: board) and *vergadering* (English: meeting). An agenda for this meeting may be called a *directievergaderingsagenda*, which is a concatenation of *directievergadering* (English: board meeting) and *agenda* (English: agenda). Such words may occur in a text as a compound but may also occur as a sequence of standalone words. Decompounding (or compound splitting) is not really an issue in English, since almost all compounds are separated by a white space (for instance: *White House*, *computer science* and *peace agreement*. Some exclusions are *breathtaking*, *fingerprint* and *whereabouts*, but such compounds are usually not represented as a sequence of standalone words (e.g. *print of a finger* for *fingerprint*).

Chen (2002) performed several experiments with compound splitting in Dutch and German in his research on Cross Language Information Retrieval. Chen experimented with a test set of 750,000 news articles categorized in 50 topics. Each topic consists of three parts: A title, a description and a narrative. For the monolingual information retrieval experiments in German, Chen used a test set of 1938 news articles categorized in 50 topics. For the experiments in Dutch, he used a test set of 1862 news articles categorized in 50 topics. For these experiments only the titles and descriptions of the articles were used. For monolingual information retrieval (MLIR) in German, decompounding caused an increase of 11.47% in average precision and an increase of 16.04% in recall. For Dutch monolingual information retrieval, decompounding caused an increase in average precision of 4.10% and an increase in recall of 3.91%. Moreover, the combination of stemming and decompounding improved the average precision for German MLIR with 26.89% and recall with 24.79%, as for Dutch MLIR, this combination improved the average precision with 10.57% and recall with 4.55%.

Monz and De Rijke (2001) show in their experiments for CLEF 2001, that decompounding in Dutch MLIR increases the average precision with 6.1% and in German MLIR with 9.6%. Like Monz and De Rijke, we use the Dutch lexicon of Celex to implement a compound splitter in the e-mail classification system. The compound splitter replaces a compound word with the standalone words the compound is built of in all documents of the collection. A (Dutch) fragment of the decompounding lists from the Celex Lexicon is given below (each word at the left side is split in the words between the brackets):

```
schoolrapport    [ school & rapport ]
schoolrapporten  [ school & rapporten ]
schoolrecht      [ school & recht ]
schoolrecord     [ school & record ]
schoolrecords    [ school & records ]
schoolregel      [ school & regel ]
schoolregels     [ school & regels ]
```

5.6 Part of Speech tagging

Part of speech (POS) tagging is the process of assigning a part of speech (also known as word class or morphological class) to each word in a corpus (or document) (Jurafsky and Martin, 2000). POS tagging is an important building block for language technology in the sense that it gives a significant amount of information about the word and its neighbours. For instance, a personal pronoun (*I* ,*she* or *he*) is usually followed by a verb (e.g. *I am*), while a possessive pronoun (*my*, *his* or *hers*) is usually followed by a noun (e.g. *my bicycle*). For information retrieval and text classification techniques, POS tagging can be used to enhance stemming accuracy (since we know the morphological affixes a word can take if we know its word class) and may help in feature selection (e.g. a noun is usually more meaningful than an adverb). The most difficult aspect of POS tagging is to cope with ambiguous words like *book*, which can be used as a noun (*a science book*) or a verb (*book me a flight to Tokio*). The result of POS tagging a simple sentence like *The grand jury commented on a number of other topics*, may look like:

```
The/DT grand/JJ jury/NN commented/VBD on/IN a/DT
number/NN of/IN other/JJ topics/NNS
```

in which tags are preceded by a slash, DT is a determiner, JJ is an adjective, NN a singular noun, VBD a verb in past tense, IN a preposition and NNS a plural noun. We can roughly distinguish two types of POS taggers: rule based taggers and stochastic taggers, which will briefly be discussed below (using Jurafsky and Martin, 2000).

5.6.1 Rule based tagging

Rule based POS tagging systems are usually based on a two-stage architecture. The first stage uses a dictionary to list all potential parts-of-speech for each word in the input sentence, while the second stage uses hand-written disambiguation rules to winnow down this list to a single part-of-speech for each word. In the next stage, a set of about 1100 constraints are applied to the input sentence to rule out incorrect parts-of-speech. For instance, a certain tag (or tag collection) may be ruled out if the concerning word is succeeded or preceded by a word from certain word class.

5.6.2 Stochastic tagging

Stochastic tagging (like a Hidden Markov Model (HMM) tagger) uses probabilities to estimate the word classes for the words of a sentence. For a given sentence, HMM taggers choose the tag with the highest probability based on the probability that a word occurs and the probability that a word from a certain word class may occur, given the previous *n* word classes (meaning that we choose the most probable tag sequence for a sentence). A bi-gram ($n = 2$) tagger chooses the the tag t_i for word w_i that is most probable given the previous tag t_{i-1} and the current word w_i, resulting in equation:

$$t_i = \arg\max P(t_j|t_{j-1})P(w_i|t_j) \tag{5.5}$$

For instance, if we us a bi-gram model, and we have to determine whether the tag for the word *race* should be noun (NN) or verb (VB) if it is preceded by the word *to* (word class TO), we can use the next two probabilities:

$$P(\text{VB}|\text{TO})P(\text{Race}|\text{VB}) \tag{5.6}$$

$$P(\text{NN}|\text{TO})P(\text{Race}|\text{NN}) \tag{5.7}$$

In equation 5.6 we assume that *race* is a verb, and in equation 5.7 we assume it is a noun where the probabilities are calculated by multiplying the probability that *race* is a noun with the probability that a noun is preceded by *to*.

Stochastic POS taggers are machine learning algorithms and should be trained before they can be used to determine the correct tags for an input sentence. Supervised learning is the most common training technique for machine learning algorithms, but also the most labour intensive one since a corpus should be manually annotated to be used as training material. However, unsupervised training of an HMM model is also possible, enabling training on unlabelled data. These taggers start with a dictionary which lists all potential tags for each word, and use the Expectation Maximization (EM) algorithm to learn a likelihood function for each tag, and the tag transition probabilities. However, experiments show that HMM's trained on manually tagged data outperform those that use unlabelled data and EM training. We refer to Jurafsky and Martin (2000) for more detailed information on Hidden Markov Models and EM training.

5.6.3 Transformation based tagging

Transformation based tagging (also called Brill tagging) is an instance of the Tranfsormation Based Learning (TBL) approach to machine learning (for more information we refer to Brill, 1995), and is based on both rule-based and stochastic tagging methods. The TBL tagger is based on rules that specify which tags should be assigned to which words, like the rule based taggers. But like the stochastic taggers, TBL is a machine learning technique in which rules are automatically induced from the data. TBL uses a set of tagging rules in the following way: The corpus is first tagged using the broadest rule (i.e. the rule that applies to most cases), then a slightly more specific rule is chosen which changes some of the original tags and next an even narrower rule is chosen, which changes a smaller number of tags (some of which may be changed previously).

Brill's TBL algorithm has three major stages. First, it labels every word with its most likely tag. It then examines every possible transformation and selects the one that results in the most improved tagging. Finally, it re-tags the data according to this rule. These three stages are repeated until some stopping criterion is reached (e.g. insufficient improvement over the previous pass). In order to complete stage 2, TBL needs to know the correct tag of each word, and is therefore called a supervised learning algorithm. Templates are used to re-tag words based on the tags of other words (like "Change tag **a** into tag **b** if the preceding word is tagged **z**").

5.7 Semantic clustering

Thus far, the language technology discussed in this thesis focussed on syntax only, but we could also consider using semantics in our classification system. Text categorization might benefit from using synonyms or semantic concepts (e.g. *motor cycle* and *car* are both instances of the concept *motorised vehicles*) (which is a hypernym of both), enabling us to relate different words with the same or similar meaning.

Rosso et al. (2004) has experimented with the use of WordNet senses as index terms, instead of just using the words in a document. WordNet[4] is a lexical database in which English nouns, verbs, adjectives and adverbs are organized into synonym sets (called *synsets*), each representing the underlying lexical concept as basic building blocks. Wordnet uses 200,000 English terms and 600,000 links to represent semantic relations between words (such as antonyms, hyponyms, hypernyms and synonyms). They found that indexing WordNet synsets instead of terms, slightly improved classification accuracy using K-NN classification (with $K = 30$), on a dataset of 20,000 newsgroup postings, divided in 20

[4]Freely available at http://wordnet.princeton.edu/

distinct newsgroups.

Gomez et al. (2004) also experimented with concept indexing using WordNet synsets, and were not able to prove that concept indexing is better than the bag-of-words approach for text categorization. For their experiments, they used the Semcor corpus, which is a subset of 250,000 words and 15 categories, of the Brown Corpus.

Chapter 6

Approach

In this section we will present the approach we have chosen to prove our hypotheses. The theoretical basis for our approach has already been discussed in chapters 3, 4 and 5. Since we claim that information retrieval based classification approaches outperform the approach Em@ilco currently uses for answer suggestion (hypothesis 1), we begin this chapter with a brief outline of the approach of Em@ilco.

6.1 Current Em@ilco approach

Em@ilco has developed the Q.mail//box to enable companies to manage their e-mail correspondence with customers. All incoming e-mail is sent to the contact centre agents who are responsible for answering it. This system also features a routine for suggesting possible answers from a Question/Answer database. For every incoming e-mail, the Q.mail//box searches suitable answers in the Question/Answer database, based on manually determined keywords. A set of keywords is defined for every standard question of the database by the content manager of the system, and every new e-mail is checked by the system on the presence of these keywords. The keywords that are found in the new mail are compared to the predefined keyword sets of every standard question and the question that has most keywords in common with the new mail, probably leads to the best suitable answer for this mail. Results are ranked in decreasing order of the number of keywords they have in common with the new e-mail. An example: if a new e-mail has four keywords in common with standard question 1, three with standard question 2 and one with standard question 3, the answer associated with standard question 1 gets a score of 100%, the answer of standard question 2 a score of 75% and the answer of standard questio 3 a score of 25%.

The main disadvantage of this method to suggest possible answers is that the keyword sets associated to each standard question are defined by a human content manager, and not by an algorithm. This could still lead to good results if the QA database is small (e.g. about 20 Question/Answer pairs), but when more QA pairs exist, it is very difficult to define accurate keyword sets that are dis-

Characteristics	**Reuters**	**Busemann**	**Our corpus**
Document type	news	e-mail	e-mail
Language	English	German	Dutch
Total nr. of documents	21,578	5,008	30,828
Used nr. of documents	13,321	4,777	16,198
Total nr. of categories	135	74	143
Used nr. of categories	120	47	37
Average document length	129 words	60 words	80 words

Table 6.1: Comparison of two e-mail corpora with the Reuters corpus

tinctive enough. Also, the ranking algorithm can be very deceptive because if, for instance, the suggested answers all have two keywords in common with the new mail, all suggestions get a score of 100%. Such a score suggests that the suggested answer must be the correct one, but this decision should not be made on two (possibly unimportant) keywords.

6.2 E-mail corpora

Busemann et al. (2000) have developed the ICC-mail system: an e-mail answer suggestion system that can be used in a contact centre environment (see chapter 2). They use a corpus of 4,777 e-mails and 74 categories (not all categories are used). They compare their e-mail corpus with the Reuters[1] corpus (containing small news documents) which is often used in benchmarking tests. Table 6.1 displays the differences between Busemanns corpus, the Reuters corpus (TOPICS test set containing 13.321 documents) and the corpus used in this research. The TOPICS test set of the Reuters corpus contains news messages in the economic domain, where 120 of the 135 categories contain one or more documents. In Busemanns corpus, a category is used if it contains at least 30 documents, which cover a total of 94% of all documents in the collection.

The e-mails available for this research have been collected in a contact centre environment for a Dutch national lottery. The collected corpus consists of approximately 28,000 e-mails, from which we can use almost 17,000 e-mails (for detailed information we refer to section 8.1.1).

As can be seen in table 6.1, these three corpora differ a lot from each other, but we can derive some useful relations between the e-mail corpus of Busemann and ours. Let us first point out the main difference between an e-mail corpus an a corpus like Reuters. The Reuters corpus (mostly) contains morphologically and syntactically well formed documents, whereas an e-mail corpus usually contains spontaneously created and informal documents, requiring researchers to cope with a large amount of jargon, misspellings and grammatical inaccuracy (Busemann et al., 2000). Besides the fact that both e-mail corpora deal with e-mail, there are also similarities between both languages, as the Dutch and German language are very alike.

[1]available at http://www.daviddlewis.com/resources/testcollections/reuters21578/

6.3 Problem analysis

In chapter 1 we presented a brief introduction on the approach we use to prove our hypotheses. First of all, we are going to cast the e-mail answer suggestion problem to a text classification problem by using the standard question/answer pairs as categories and the new and previously answered e-mails as documents. If we are able to categorize a new e-mail message in one of the question/answer categories, we can use the corresponding answer as a suggestion. The use of text classification methods for e-mail answer suggestion introduces three important aspects we should keep in mind when we choose an approach to follow.

Because e-mail is such an easily accessible means of communication, e-mails are often unstructured and informal texts in which spelling errors and grammatically inconsistencies occur quite often.

Moreover, some people may pose their question straight out because they know what information they want, resulting in very short and accurate messages. However, other people may describe their problem or question very elaborately, because, for instance, they cannot pinpoint the cause of the problem, or are incapable of describing the problem or question briefly. Such large e-mail messages may contain lots of information that does not help at all in automatically finding the correct answer, since overlap with other categories may occur due to the wide scope of words that are used. Therefore, we have to consider the varying message length in choosing our approach.

The last important aspect of this problem is the level of detail of the classes we wish to categorize our e-mail in, with respect to a document classification system in which topics of the documents are used (e.g. sports, finance, culture, etcetera). In contact centres like the one used in this thesis, distinct questions about the same topic may be posed: for instance a customer may want to cancel all tickets (stop the lottery), or may want to cancel just 1 ticket (both questions are about the topic "cancelling lottery tickets").

6.4 Classification

The idea behind this research was that we should be able to provide relevant answer suggestions to incoming e-mail, based on similar e-mail that has been previously answered. We stated that if we could find similar messages (i.e. like finding relevant documents using a search engine), we could use these in order to select a relevant answer. Such IR-based approaches have been extensively investigated for text categorization using the Rocchio classifier (section 4.3.1) and K-Nearest-Neighbour classification (section 4.3.2).

In chapter 4 we also discussed two other classification approaches that perform good on text classification problems: Naive Bayes (4.1) and Support Vector Machines (section 4.2). Naive Bayes tries to estimate the probability that a certain feature vector belongs to a given category, based on the probability that all distinct features from the vector (words in a document) belong to a given category. Support Vector Machines learn a threshold function to separate data in a high-dimensional space using an error minimization algorithm. Since Naive Bayes and Support Vector Machines have already been proven suitable for e-mail answer suggestions (respectively by Gaustad and Bouma (2002) and Scheffer (2004)), and we would like to pursue our hypothesis that answers could also be suggested using information retrieval based techniques, we will try to classify e-mail using the Information Retrieval Based classification approach. Within this IR-based approach, it would be interesting to see if a profile based classifier (like Rocchio) or an example based classifier (like K-NN) performs best on this specific classification problem. Yang (1999) showed that the K-NN classifier outperforms the Rocchio classifier, and the Naive Bayes classifier on the Reuters dataset, with 85% accuracy for the K-NN classifier, to 75% and 71% for Rocchio and Naive Bayes respectively, on version 3 of the Reuters dataset.

The Rocchio classifier builds a profile for each category and tries to match new messages to these profiles. Such a profile is a unique representation consisting of keywords that are specifically interesting for the category which is represented by the profile. Yang (1999) explains that the weakness of the Rocchio classifier is the assumption of one centroid (profile) per category, which is why Rocchio does not perform well when the documents belonging to category naturally form separate clusters. Most likely, this is not the case in our email classification problem, since the documents in a category are all about the same question.

The K-NN classifier tries to classify a new message by finding the K most relevant messages that have previously been answered. From these K most relevant documents, the similarity scores for each corresponding category are summed, and the category with the largest sum is the best suggestion.

For the IR-based classification approach we will investigate the use of the TF.IDF weighting scheme, the Okapi weighting scheme and the Cosine and Inner Product similarity measures. Both combinations Okapi/Inner Product, and TF.IDF/Cosine have normalization routines for document length, which can be used to compensate for the difference in message lengths in our e-mail classification problem.

6.5 Language Technology

Language technology can be used as a text normalization process prior to the classification process. Using language technology, we try to compensate for the lack of structure in e-mails and the possibility of grammatically incorrectness and spelling errors. For our language technology experiments we use the Lingware tool-kit implemented by Carp Technologies. For this research they have kindly made available a tool-kit which incorporates stemming, part-of-speech tagging and spelling correction.

6.5.1 Lexical normalization

The first process in which we use (very simple) language technology is lexical normalization. In this step we try to normalize the e-mail by removing unwanted characters and strings. Because in typing text the use of diacritics (like *é* or *ñ*) is somewhat laborious, most people tend to omit them. Therefore we translate them to the same character without diacritics (e.g. *é* becomes *e*). Because we are working with an e-mail corpus consisting of questions about a lottery, we opted to neglect the use of numbers. Because the lottery tickets use zip codes, parts of these numbers are not unique, but have a very distinctive power (resulting in a high relevance weight). We assume that the use of these numbers is independent of the question that is asked (lots of customers might mention the lottery ticket number, but it is not needed for determining the question) since we do not want a document to receive a high relevance score just because of this ticket number.

Some of the incoming e-mail messages are submitted using a web form. This web form is then e-mailed to the contact centre, and uses a standard layout with meta-information about the message. This meta-information (like the date it is sent) and the layout tags are removed from the e-mail message since they do not attribute to the intentional meaning of the message. Also, e-mail addresses are removed from the original message, because classification based on the occurrence of an e-mail address might cause a problem if the customer has sent multiple e-mails, containing different questions). Finally, we remove all non-alphabetical characters, because they might disturb the tokenization process and transform all characters to lower-case to ensure equal representation of equal words. This results in a representation of each word that may only contain lower-case characters ('a',...,'z').

After the lexical normalization, the input is sent to the Lingware tool-kit. The Lingware tool-kit performs best if it receives full sentences as input, for disambiguation using Part-of-Speech tagging (see section 5.6). The tool-kit also provides routines for stemming and spelling correction. The last stage is to tokenize the output of the tool-kit, resulting in a bag-of-words (BOW) representation for each document.

6.5.2 The use of language technology for classification

The use of language technology in IR and classification is two-fold. Language technology like POS-tagging, Language identification and stopword removal are feature selection routines. The nature of these technologies is to eliminate words that are not relevant for the classification of a document and provide a reduction in the feature space. By eliminating non-relevant words (like stopwords or prepositions), the classification process may focus on the relevant words only and may result in a higher classification accuracy. Language identification can also be used as a feature selection technique, by eliminating all words that do not belong to a language. The danger of applying such feature selection techniques is that we accidentally may eliminate words that actually were important (or relevant) for the classification process.

The second reason to use language technology has a more statistical nature. In classification (and IR) we usually base the similarity of documents on the words that occur in them. However, a document in which the word *House* occurs ten times, may be just as relevant to a query as a document in which the word *Houses* or *Home* occurs ten times. Our classification approaches are only able to relate terms that are syntactically equal. To compensate for the incapability of these classification approach to relate semantically equal, but syntactically different terms, we can use language technology like stemming, spelling correction, decompounding and semantic clustering. All of these language technologies determine a basic representation of different morphological variants of equal or related words. For example, the words *swimming* and *swum* can be represented as the basic form *swim* by applying stemming. Semantic clustering might relate the words *cows* and *cattle* (which is a hypernym of *cows*). Spelling correction also maps different morphological variants of a word to a basic representation (i.e. the correctly spelled word). The benefit of applying such language technologies is that more documents may be found relevant, which results in more candidates for the classification process (which is based on relevant documents). However, applying such techniques not necessarily results in a better classification result. By applying these techniques we substitute a group of words (different morphological variants or semantically related words) with a more general representation. In IR this results in a higher recall, but usually at the cost of precision. In a classification problem, we may argue that the representation of such words has become to general, which hinders the classification process because the small differences between relevant and irrelevant documents may diminish. In chapter 8 we will experiment with the application of such language technologies to determine if they improve the classification results.

6.5.3 Stopword removal

Stopwords are meaningless words that usually occur very often in texts (some exceptions are *accordingly* and *whomever*, which do occur that often). Because these stopwords do not carry meaning, they are not important for the classification process. The fact the most stopwords occur in many documents, causes their assigned relevance scores to be rather low, which makes the impact of such words on the classification process relatively small. However, the influence of stopwords on the classification process should not be neglected, because the relevance score is not equal to zero. Removing stopwords has the benefit that it reduces the feature size and in doing so, speeds up the indexing and classification process.

6.5.4 Language identification

Using N-grams (see section 5.4) we can estimate the probability that a word is part of a certain language, based on character sequence of the word and common character sequences in a given language. Normally, language identification is a text classification problem in which documents should be categorized in one of a set of languages, based on the largest probability. Our e-mail classification problem uses just one language, so we cannot really compare the results amongst a set of languages. Therefore, the probability that a word is part of the given language should exceed a certain threshold value.

In our classification system we use a 5-th order Markov Chain (the occurrence of a character depends on the preceding 4 characters) with a threshold of zero. The threshold implies that if a certain sequence of 4 characters of a word does not occur in our language corpus, we classify it is as not being Dutch[2]. The language corpus is based on the 800,000 most frequent words from the "Corpus Gesproken Nederlands" (English: Dutch Spoken Corpus, Oostdijk, 2000).

6.5.5 Part-of-speech tagging

Part-of-speech (POS) tagging has proven to be very useful in language technology and information retrieval, due to its use for disambiguation of terms. Equal words (e.g. *bank*) may have multiple meanings based on the word class (e.g. as a verb *to bank* has a different meaning than the noun *bank*). In the e-mail classification problem we use POS tagging for disambiguating words before they are stemmed and to select features (words) from documents. Words from some word classes carry more meaning than words from other word classes. Words from open word classes[3] (nouns, verbs, adjectives and adverbs) carry more meaning than words from closed word classes. Kraaij and Pohlmann (1996) stated that the majority of the successful query terms for an IR system in a collection of newspapers are nouns (58%), followed by verbs (29%) and adjectives (13%), while other categories are negligible.

For feature selection using the POS tagger, we only use nouns, verbs and adjectives, following the findings of Kraaij and Pohlmann. The POS tagger we use is implemented in the Lingware tool-kit and may be called an "unsupervised transformation based tagger". Such a tagger (described in section 5.6) is error driven, uses both stochastic and rule based tagging methods and does not need a manually pre-tagged corpus for learning.

6.5.6 Stemming

Recall from section 5.2 that stemming replaces different morphological variants of a word with the stem of that word. If a set of documents are all about the same topic (or pose the same question in our problem), but use different morphological variants (like *swimming*, *swum*, *swam* and *swim*), a classification method is unable to relate the documents based on these terms. If we apply stemming, all documents from this set now contain the same morphological variant of that word (i.e. *swim*) and can be related. Gaustad and Bouma (2002) found that stemming did not improve classification accuracy for the Naive Bayes classification method for e-mail classification. However, since stemming has proved itself useful in other text-classification approach, and moreover, we are using different classification methods, we will investigate the influence of stemming on the classification accuracy.

Since dictionary based stemming yields best stemming accuracy (Gaustad and Bouma, 2002), we decided to follow this approach. The stemming routine in the Lingware tool-kit uses a dictionary for stemming. If a word could not be found

[2]The probability that such a sequence belongs to a given class is 0, causing the probability that a word is from that class is also 0. Because of this, no smoothing techniques are used

[3]Open word classes are classes that do not have a fixed membership: new words may be added (for example by borrowing them from other languages)

in the dictionary, the stemming routine uses similar words (i.e. with the same ending and word class) for which the stemming procedure is known, and applies the same procedure to the unknown words.

6.5.7 Spelling correction

E-mails may contain spelling errors and typos, which obviously does not help in retrieving an e-mail, let alone classifying it, based on the correct spelling of the word. Ideally, we would process an e-mail in such a way that all spelling errors are corrected. Unfortunately, spelling correction is a very difficult process, which may introduce more errors than it corrects, resulting in a decrease of accuracy for the classification methods. For instance, if a misspelling is very common in a certain language, e.g. *actress* is written as *acress*, chances are that many e-mails contain this specific misspelling if the intention was to write *actress*. Since many e-mails contain this misspelled word, and some of them have been manually classified in the correct category, the classifier may have learned to classify this misspelled word in the correct category. However, if a spelling correction routine changes the spelling to *across* (both words have a Levenhstein distance of 1), the classifier has much more difficulty with categorizing the document, because *across* is a very frequent word. The biggest weakness of spelling correction routines is that they may correct words that are not misspelled (e.g. proper nouns or jargon), but just do not occur in the dictionary. In the case of proper nouns or jargon, this probably causes a decrease in classification accuracy, since these words tend to be very important for distinction of documents, and therefore for the classification process.

We are going to investigate the influence of spelling correction on the classification accuracy using the Lingware tool-kit. This tool-kit has a routine for spelling correction, based on N-grams (thus it is context based), Levenhstein distance and models of common made typing errors.

6.5.8 Decompounding

Decompounding or compound splitting, is a specific language technology often very useful for compounding languages like Dutch, German or Finnish. The possible usefulness of compound splitting in a Dutch information retrieval or classification system, is that such compounding words may be written as one whole (*directiesecretaresse*, English: secretary of the board) or as a series of standalone words (*secretaresse van de directie*). An IR or classification system that does not use semantics is not able to relate these two terms, unless the compound is split.

As described in section 6.5.2, splitting compounds does not necessarily result in better classification results. Suppose a customer poses the question *"Ik probeer via de telefoon iets te vragen over mijn internetverbinding"* (English: I am trying to pose a question via the telephone about my internet connection) and another customer poses the question *"Ik probeer via internet iets te vragen over mijn telefoonverbinding"* (English: I am trying to pose a question via the Internet about my telephone connection). If we split the compounds *telefoonverbinding* and *in-*

ternetverbinding, a classification routine is unable to detect the difference between the questions (since the BOW representations of both questions are equal).

The decompounding routine we use in this research, uses the Celex lexical database. If a compound is listed, we replace it by the remaining word parts.

6.5.9 Semantic clustering

Semantic clustering may assist in relating messages based on semantics, instead on the syntax of the messages. If a customer wishes to receive information about motorised vehicles, we may present him or her with documents that contain the words *motorised* and *vehicles*. However, this user is probably also interested in cars and motor cycles (both are motorised vehicles), which may not be covered in the presented documents about motorised vehicles. *Cars* and *Motor cycles* are both instances of the concept *motorised vehicles* (which is a hypernym of both). The use of semantic clustering may assist in detecting such relations between words and create a mapping from these words to the parent concept (motorised vehicles in our example). Instead of the individual words of the document, we may use the concept of the words in the classification process. The advantage of applying this technique is to generalise the content of the documents to find more similar (and relevant) documents. However, semantic clustering is a considerable generalisation process, which may hamper the process of categorization.

For Dutch, a semantic lexicon is available (EuroWordNet), but unfortunately, the licenses to use it are expensive. However, we believe that using semantic concepts as index term, we overgeneralise the e-mails. Because the differences between the categories are very small, using semantic relations (synonyms and hypernyms) hamper the process of categorization. Besides, Rosso et al. (2004) and Gomez et al. (2004) did not find a significant improvement in classification accuracy on a classification problem in which the categories are more general.

Chapter 7

Design

Since no text classification system containing all the aspects of our approach was available, we decided to implement such a system ourselves. This chapter gives an overview of the design, and presents the algorithms that constitute the foundation for this system. Additionally, it presents an example of the indexing and classification processes, providing a better understanding of these processes.

7.1 System overview

The e-mail classification system can be divided into three parts: document preprocessing, indexing and categorization. In order to classify incoming e-mail, reference material is needed to which the new mail can be compared to. This reference material (the previously answered mails) is analysed by the document preprocessing subsystem and all useful tokens (words) are stored in an index called the Document Collection. This index stores all index terms accompanied by the following information:

- The documents the term occurs in
- The weights associated to the term/document pair

The indexing subsystem creates such an index using the output form the document preprocessing subsystem, which is a bag-of-words for every document in the collection. Figure 7.1 shows the architecture of the document preprocessing and indexing phase for documents as figure 7.4 shows the architecture of the document preprocessing phase in combination with the classification phase.

7.2 Document preprocessing

In this subsystem each query document (incoming mail) and training document (previously answered mail) is analysed and transformed into a bag-of-words rep-

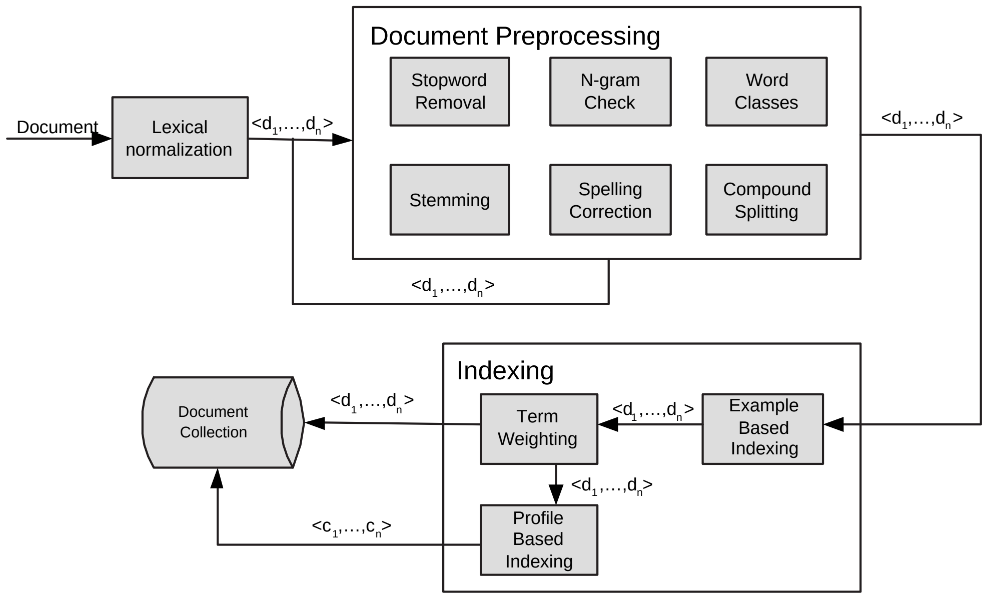

Figure 7.1: The indexing procedure: First a document is lexically normalized and language technology may be applied. The indexing subsystem creates an index of document representations (for the example based classifier) and an index of category representations (for the profile based classifier).

resentation. Figure 7.1 displays the document preprocessing phase for indexing training documents and figure 7.4 for the classification phase. Both preprocessing phases differ only on the input: in figure 7.1 the input is a training document and in figure 7.4 the input is a query document. The document preprocessing subsystem contains six language technology processes, that are already described in chapter 5. A bag-of-words may be processed by the subsystem multiple times, enabling us to use more language technologies, in any desired order.

Recall from chapter 6 that the Lingware tool-kit uses complete sentences for its best results, instead of a bag-of-words representation. In our implementation, we have taken this requirement into account by sending complete sentences to the tool-kit. The full-sentence representation is an intermediate representation between the full text and the bag-of-words, in which punctuation marks and diacritic marks have been removed. After the Lingware tool-kit has been used, we transform the intermediate sentence representation to the bag-of-words representation.

7.3 Indexing

The indexing procedure is probably the most important procedure in the classification system. If an index is well created, it allows us to find relevant documents to a query as fast as possible. In order to classify new documents (incoming e-

Document	Words	Category
Doc1	*bicycle, car*	1 (land vehicles)
Doc2	*bicycle, car, car*	1 (land vehicles)
Doc3	*boat, boat*	2 (water vehicles)
Doc4	*airplane, airplane*	3 (air vehicles)

Table 7.1: A small document collection

mail), an index should be created containing all words that can be found in the training documents (previously answered e-mail) after document preprocessing. This index can be compared to an index that is normally found at the end of a book and stores per index term the documents it occurs in and the weight that is assigned to every term/document combination. Such an index is called an inverted file index. As discussed in chapter 6 we investigate two different classification approaches (example and profile based) using two different weighting schemes (Okapi and TF.IDF). For each classification approach, a different indexing procedure is needed. As well the Okapi as TF.IDF weighting scheme are suitable for term weighting in both classification procedures.

In figure 7.1 documents and categories are represented by a BOW representation: a document is denoted with $(d_1, ..., d_n)$ and a category is denoted with $(c_1, ..., c_n)$.

7.3.1 Example based indexing

The example based classification procedure uses information retrieval to match a new document to a set of training documents. The index for this procedure therefore consists of a collection with a representation for each document that is used for training. The output of the document preprocessing subsystem (a bag-of-words) serves as the input for the example based indexing procedure. This procedure creates an index based on the words (index terms) accompanied by the following information:

- In which and how many documents it occurs in (document frequency), along with the document length
- The number of times it occurs in each document (term frequency)

The algorithm to index the words in a document collection is shown in figure 7.2. In this algorithm we list all the words in a document collection and record the term frequency for each term/document pair.

Suppose we want to create an index of the small document collection shown in table 7.1. These four documents contain four unique terms and are categorized in three distinct categories (land, water and air vehicles). Table 7.2 shows the indexing of this document collection containing four documents and three categories. The first step shows the result after listing all term/document pairs that are encountered (this list is already sorted). The second step shows the results after grouping identical index term/document pairs. The entry

1. For each document:
   List each word that occurs in it and the document name
2. Sort the list alphabetically and iterate over it
3. Group identical index term/document pairs and record the term frequency for every pair
4. Group identical index terms and record the document frequency for every index term
5. Finally: Write all the index term/document pairs to the index

Figure 7.2: Example based indexing algorithm

Step 1		**Step 2**		**Step 3**	
Term	**Doc**	**Term**	**Posting**	**Term**	**Posting**
airplane	Doc3	*airplane*	(Doc3, 2)	*airplane*	(Doc3, 2)
airplane	Doc3	*bicycle*	(Doc1, 1)	*bicycle*	(Doc1, 1) ; (Doc2, 1)
bicycle	Doc1	*bicycle*	(Doc2, 1)	*boat*	(Doc4, 2)
bicycle	Doc2	*boat*	(Doc4, 2)	*car*	(Doc1, 1) ; (Doc2, 2)
boat	Doc4	*car*	(Doc1, 1)		
boat	Doc4	*car*	(Doc2, 2)		
car	Doc1				
car	Doc2				
car	Doc2				

Table 7.2: Example based indexing

Boat (Doc4, 2)

means that index term *boat* occurs in document *Doc4* with term frequency 2. The third step displays the final results of the example based indexing procedure. Every index term occurs only once in the index, followed by the list of documents in which it occurs. This list of documents is called the posting of an index term. As we can see in table 7.2, index term *car* occurs in two documents (once in *Doc1* and twice in *Doc2*).

Term weighting

To improve the results of the information retrieval routines, all index term/document pairs should be weighted. The more important and distinctive an index term is in the document collection, the higher the weight associated to it. Recall that we investigate two different term weighting schemes. The data for calculating each weight can be derived from the example based index. For each index term/document pair in the index a weight is calculated. For the TF.IDF weighting scheme the weight is based on the term frequency and the (inverse) document frequency, as the weights for the Okapi weighting scheme are also based on the document length, average document length and the number of documents in the collection.

Following the example of table 7.2, we can calculate the weights for these index terms using the TF.IDF weighting scheme. Recall from section 3.3.1 that a weight d^i is calculated using the term frequency and inverse document frequency. The term frequency for index term *car* in document *Doc2* is 2. Since *car* occurs in two documents, and the total document collection contains four documents, the TF.IDF weight for this term is calculated following equation 7.1. For the convenience of the equation we have used a base of 2 for the logarithmic function. In the weighting algorithms, the logarithm may be taken to any convenient base.

$$d^i = TF \times IDF(w_i) = \quad log_2(\frac{|D|}{DF(w_i)}) = \quad 2 \times log_2(\frac{4}{2}) = 2 \tag{7.1}$$

For indexterm *boat*, occuring only in document *Doc4* (with term frequency 2) the weight is calculated as shown in equation 7.2:

$$d^i = TF \times IDF(w_i) = \quad log_2(\frac{|D|}{DF(w_i)}) = \quad 2 \times log_2(\frac{4}{1}) = 4 \tag{7.2}$$

The first two columns in table 7.3 show the example based index using the TD.IDF weighting scheme for our small document collection. The term frequencies for index terms in each document are replaced by the TF.IDF weights as calculated in both examples above.

Term	**Posting**	**Term**	**Posting**
airplane	(Doc3, 4)	*airplane*	(Cat2, 16)
bicycle	(Doc1, 1) ; (Doc2, 1)	*bicycle*	(Cat1, 9.235)
boat	(Doc4, 4)	*boat*	(Cat3, 16)
car	(Doc1, 1) ; (Doc2, 2)	*car*	(Cat1, 12.812)

Table 7.3: Indexing results

7.3.2 Profile based indexing

The example based classification approach was based on the similarity between the query document representation and the representations of the documents in the collection. However, the profile based classification approach is based on the similarity between the query document representation and the category representations. For each category, a representation is calculated by taking all positive and negative examples for this category. A positive example is a document that should be categorized in the category and a negative example should be categorized in another category. Since the profile based indexing procedure requires a weighted index as a starting point, this index is based on the example based index explained above. Figure 7.3 displays the profile based indexing algorithm. The normalized weight of an index term is simply the calculated weight divided by the Euclidean document length of the document it occurs in. The sum of the normalized weights of all terms in a document should be 1.

The results of this indexing procedure are similar to the results of the example based indexing procedure, with the difference that we now store the (normalized) weight per index term/category pair instead of the weight per index term/document pair. Suppose we continue indexing the example document collection of table 7.1. This collection is set up in such a way that a specific index term occurs in only one category (leaving us with only positive examples for calculating the profile weight for an index term/category pair). First, we calculate the total normalized weights per index term. The Euclidean document length of document *Doc2* is calculated by taking the square root of the sum of squares of the weights of the index terms in *Doc2*, in this case: $\sqrt{1^2+2^2} = \sqrt{5}$. The total normalized weight for index term *bicycle* can now be calculated by adding the normalized weights of *bicycle* for each document this term occurs in. *Bicycle* occurs in documents *Doc1* and *Doc2* with weight 1, the Euclidean document lengths are $\sqrt{2}$ and $\sqrt{5}$ respectively, resulting in a total weight of:

$$\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{5}} \approx 1.154 \tag{7.3}$$

Since *bicycle* occurs only in category 1, the score in equation 7.3 is already the total normalized score for this index term/category pair. If *bicycle* would occur in other categories, then the weights can be calculated using the total normalized weight of an index term and the total normalized weight per category of an index term. *Bicycle* occurs in two documents in category 1, and no documents that contain the word *bicycle* belongs to another category, so we can calculate the profile weight for this index term in category 1 by using only the first half of equation 4.2 (section 4.3.1)[1]:

$$\vec{c}_j = \alpha \left(\frac{1}{|C_j|} \sum_{\vec{d} \in C_j} \frac{\vec{d}}{\|\vec{d}\|} \right) = 16 \times \frac{1}{2} \times (\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{5}}) \approx 9,235 \tag{7.4}$$

The weights for the other index term/category pairs are calculated in a similar way. If negative examples for a category exist, for instance when the index term *boat* should occur in categories 1 and 2 because there are boats that can also move on land (e.g. a hovercraft), then the second part of the equation is also required. Calculating the total normalized weight for negative examples is done in a similar way to that of the positive examples, but now all the index terms that do not occur in a category should be taken into account. Columns 3 and 4 of table 7.3 display the results of the profile based indexing procedure on the example document collection. The entry

[1]In this equation we use $\alpha = 16$, following the literature on Rocchio classification.

```
1.   List all the index terms alphabetically and iterate over it

2.   For each index term:
     - Calculate the total normalized weight per index term

3.   For each index term/category pair:

3a.  - Calculate the normalized weight per pair

3b.  - Calculate the total normalized weight for all other
       index term/category pairs using steps 2 and 3a

3c.  - Calculate the category weight for this index term/category
       pair using equation 4.2 (section 4.3.1)

4.   Calculate the category lengths by adding all weights per category
```

Figure 7.3: Profile based indexing algorithm

car (Cat1, 12.812)

means that index term *car* is represented in category 1 (land vehicles) with weight 12.812).

7.4 Classification

Once the index is created, we can retrieve all documents (or categories) containing certain words. This process is called information retrieval. The occurrence of a word in a certain document may be far more important than the occurrence of that same word in a document that is ten times as long. For example, if the term *Clinton* occurs four times in an article and ten times in the whole news paper, it is far more important for the (small) article than for the whole news paper. Therefore, a ranking of the retrieval results is created, so the results can be processed in descending order of relevance. This ranking is the basis of the classification process implemented in the e-mail classification system. Figure 7.4 shows the implementation of the classification and document preprocessing subsystems. Note that the document preprocessing subsystem is the same as in figure 7.1 but that the input is a query document (or just a query) instead of a training document. The query is preprocessed by the document preprocessing subsystem and enters the classification subsystem as a bag-of-words represented by $< q_1, ..., q_n >$.

For the Okapi weighting scheme, experiments have been done with term weighting for longer queries. Therefore the first step of the classification procedure is to assign weights to a query. These Okapi weights are calculated using equa-

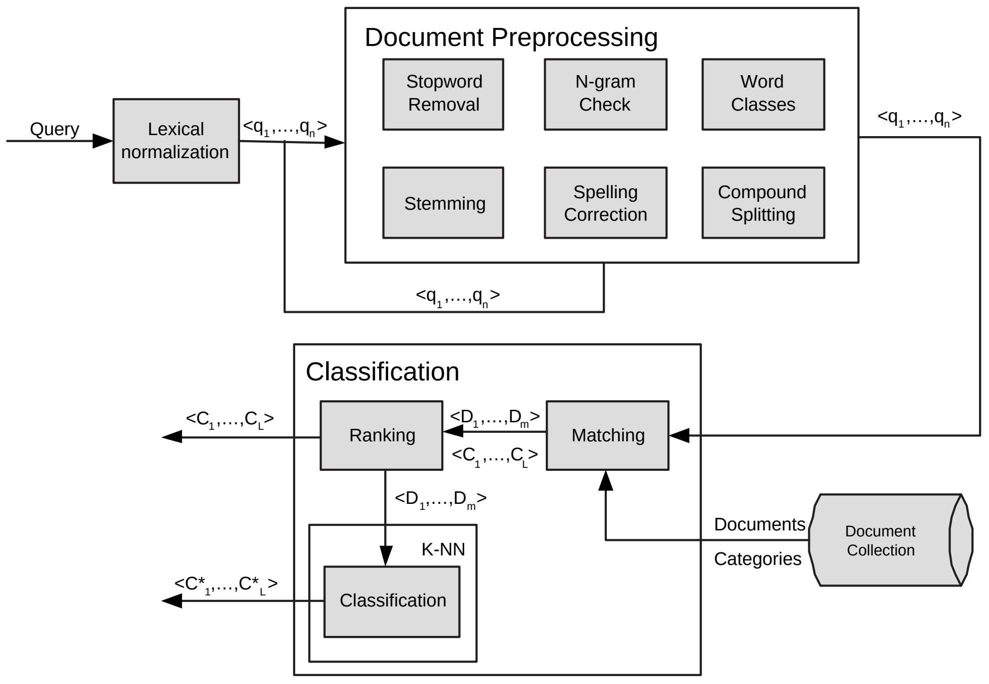

Figure 7.4: The classification procedure: The classification subsystem matches the query with all document representations (example based) and all category representations (profile based). Finally the results are ranked and outputted.

tion 3.16 in section 3.3.2. The most important step is the matching procedure. This procedure evaluates the query against all documents (for the example based classification) or categories (for the profile based classification) and returns all documents or categories of which one or more words match the query.

7.4.1 Profile based classification

The profile based classification procedure classifies documents based on category representations build by the profile based indexer (section 4.3.1). The incoming query is compared to all the category representations based on one of the matching equations (the inner product or cosine similarity measure) by the matching procedure. All categories that contain words which are also in the query, are nominated. This set of categories that match the query are sent to the ranking procedure that ranks the categories in descending order of relevance. The relevance is determined by the similarity score that is calculated by the matching procedure. The higher the similarity score, the more relevant the category is. The output of the classification subsystem for profile based classification is a ranked list of categories ($< C_1, ..., C_L >$) sorted in descending order of relevance.

7.4.2 Example based classification

The example based classification procedure classifies documents based on other (training) documents. The matching procedure is similar to the matching procedure of profile based classification, but in this case the query is compared to all documents of the document collection (using the inner product or cosine similarity measure). If a document contains words that occur in the query, it is nominated and the set of nominated documents is sent to the ranking procedure which ranks the documents in descending order of relevance. Where the profile based classification was done at this point, the ranked document ($< D_1, ..., D_m >$) set is sent to the K-NN classification procedure. This procedure uses equation 4.6 (section 4.3.2) to calculate the score for each category a nominated document occurs in, based on the first K documents from the ranked results. This procedure sums up all the weights of documents (within the first K ranked documents) that belong to the same category and reranks the category results. The output of the classification subsystem for example based classification is also a ranked list of categories ($< C_1^*, ..., C_L^* >$) sorted in descending order of relevance.

7.5 Data storage

If the e-mail classification system is operational, it should classify several e-mails per minute. It is therefore crucial that the performance of the classification subsystem and document preprocessing subsystem is guaranteed to be good enough to ensure this. To ensure fast classification, we have chosen to use a database management system to store the indexes. Such a database system allows us to retrieve data very fast and with SQL queries we can let the database do a lot of our work by grouping results and adding weights of the results of an e-mail that has to be classified.

Chapter 8

Evaluation

In this chapter we will discuss the experiments we have conducted. In the first section we will present the experimental set-up, and in the remaining sections we will present a selection of the results.

All experiments in this section have been conducted using the e-mail classification system developed for this research and the test set described in section 8.1.1. The approach for these experiments is described in chapter 6.

8.1 Experimental Set-up

8.1.1 E-mail Corpus

In order to evaluate our e-mail classification system, we need a huge amount of human categorized e-mails for training and testing. The e-mail corpus we conduct our experiments on, is collected in a contact centre of "De Nationale Postcode Loterij" (NPL, English: The National lottery of zip codes). In this national lottery, customers use their zip codes as lottery tickets. The contact centre receives e-mails with general questions about the lottery, change of addresses, technical questions about personal web pages of customers, incorrect invoices and common complaints. The e-mails collected in this contact centre have (mostly) been answered using standard answers (that can be selected from a tree-structure). The standard answer is considered as the category of the e-mail. Therefore, all e-mails that receive the same answer are considered as belonging to the same class. This e-mail corpus consists of 30,828 e-mails categorized in 143 categories. For training purposes we can not use empty categories, or categories with just a small number of mails, therefore we only use categories that contain at least 100 e-mails: resulting in an initial test set of 50 categories that contain a total number 27,789 mails.

Unfortunately, during the analysis of this initial test set, it became clear that this set was not suitable for training a classification system due to the many classification errors that had been made in the contact centre. The explanation for these errors is that the agents in the contact centre are not obliged to select a standard answer, but may also formulate one themselves (i.e. if they believe no standard answer fits). As a result, the suggested standard answer and thus the category, remains glued to the e-mail although the final answer may be completely different. This e-mail corpus has clearly not been collected for research purposes, but for debugging and quality analysis purposes of the e-mail management system developed by Em@ilco. Therefore, we first had to make a selection of usable categories for evaluating the mail classification system. The adjustments listed below have been made to the initial test set of 27,789 mails and 50 categories:

- 2 categories were excluded because the messages they contained could not be answered using a standard answer. These categories were used to store mails that could not be categorized (usually resulting in an answer like: "We could not answer your e-mail because we do not have your personal data"),
- 6 categories were excluded because they contained too many classification errors made by the agents in the contact centre.
- a set of 8 categories turned out to be a subset of 3 categories. The mails in the 8 subsets could all be answered using only 3 standard answers. The categories that require the same answer are merged, resulting in 3 (collective) categories.

These adjustments to the initial test set led to the final test set containing 16,798 e-mails that are classified in 37 distinct categories. The average number of e-mails per category is 454, but figure 8.1 shows that the smallest category contains only 106 e-mails and the largest category contains 3593 e-mails.

8.1.2 Evaluation metrics

To evaluate classification algorithms in order to see how well they correctly classify e-mails, we use the classification accuracy as a measure (in this chapter we may simply call it *performance*). The e-mail classification system is designed for use in a communication service centre, where the performance of the system is measured by the number of correctly suggested answers. If the system suggests the correct answers (categories) for half of the documents, the performance simply is 50%. To provide assistance for the e-mail answering agents, we can suggest a number of possible answers in a decreasing order of relevance (where the most relevant answer is placed at the top). If the suggested set of answers is relatively small, but contains the correct answer, the agent is able to reply to e-mail more efficiently.

To evaluate the usability of this classification approach for automatic answer suggestion, we slightly adjust the definition of performance to best-n performance. The best-n performance is the percentage of documents where the correct category is suggested within the first n answer suggestions for this document. Suppose y_i is the correct answer (i.e. category) for document i, x_i is the set of

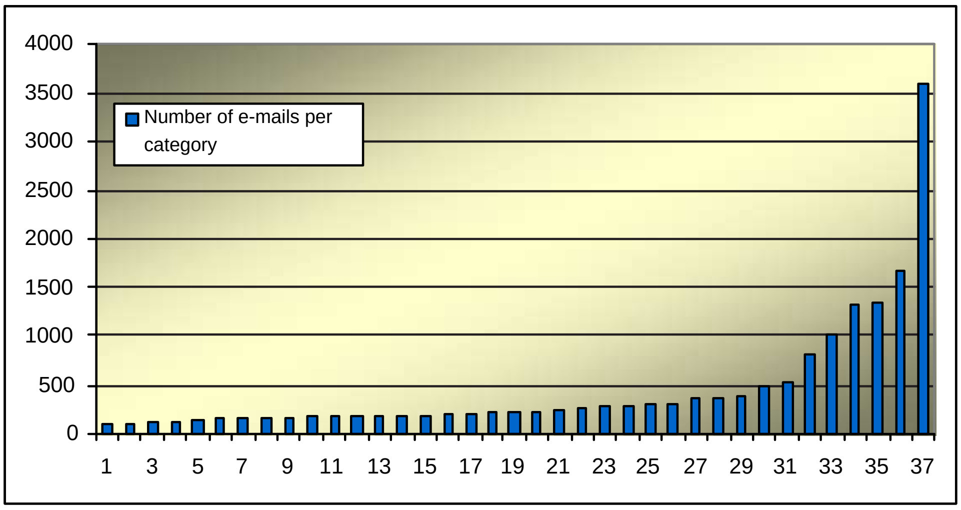

Figure 8.1: Distribution of the number of e-mails over the set of categories. The largest category (nr. 37) contains 3593 messages.

suggested documents for document i with n elements (categories), and $|D|$ the total number of documents in the collection, the best-n performance can be defined as:

$$\textit{Best-}\boldsymbol{n}\ performance = \frac{\sum_{i=1}^{|D|} f(y_i, x_i)}{|D|} \times 100$$
$$\text{where } f(y_i, x_i) = \begin{cases} 1 & \text{if } y_i \in x_i \text{ (with } |x_i| = n) \\ 0 & \text{otherwise} \end{cases} \tag{8.1}$$

Suppose we receive 100 new e-mails. For 60 of these e-mails, the correct answer is suggested within the first 5 suggestions of the system for each new e-mail. In that case, the best-5 performance of the system simply is 60%, meaning the agent can select the correct answer in 60% of the e-mails from a suggested set of only five possible answers.

8.1.3 Test set-up

As discussed in the research hypotheses in section 1.2 and the approach in chapter 6, we focus on the classification of e-mail messages using information retrieval (IR) based classification methods in the first place, and on using language technology to improve these classification results in the second place. The first experiments aim at proving that these IR based classification methods indeed outperform the classification methods used by Em@ilco, and to relate these results to other classification techniques that have been used for e-mail classification (see chapter 2), while the second series of experiments aim at showing the influence of the used language technologies on the classification results.

Training and testing

In order for our system to be able to classify e-mail messages, it has to be trained using a humanly classified subset of the NPL set discussed in section 8.1.1. These e-mail messages are used by the system as examples of each category a new message can be classified in. We use 80% of the total test set to train the classification system and the remaining 20% to evaluate the system. The evaluation is performed by feeding the classification system the messages from the test set, and let the system classify these messages in the predefined categories. Since we know the correct category for each of these test messages, we can determine if the system was able to classify the mail in the correct category, and at which rank this message was classified correctly (e.g. if the fifth category of the systems results is the correct one, the rank of this message simply is 5).

Cross validation

Randomly dividing the corpus in 80% for training and 20% for testing purposes may result in a particularly good or bad test set of e-mails, for which the results may seem more promising or demotivating than they really are. To ensure a representative test set and experiment, we use 5-fold cross validation. Using 5-fold cross validation implies making 5 distinct sets of equal size of randomly selected messages. Each message may only be selected once and thus occurs in only one of these five sets. Each set covers 20% of the total test set. With these five randomly selected sets, we conduct a series of five experiments, each time using one set as testing material and the other four as training material. The results of these five experiments are averaged, ensuring a representative experiment. In a series of five experiments, each document will be used exactly once for testing, and four times for training.

8.1.4 Parameter settings

In some of the algorithms we use for our classification problem, we may change the parameter settings. In this section we will discuss the parameters settings for each of the algorithms, in order to achieve an optimal classification accuracy.

Term Weighting

For the Okapi term weighting scheme (equation 3.14, section 3.3.2) we can adjust the influence of the term frequency by setting parameter k and the effect of the document length by setting parameter b. Robertson and Sparck Jones (1997) explain that b can be assigned a value between 0 and 1. Assigning $b = 0$ represents the assumption that documents are long because they cover multiple topics, in which case the document length should not be used as a normalization for the relevance term (since the description of each topic is relatively small). If we assign $b = 1$ we assume that documents are long because they are repetitive, and cover the same topic multiple times. Robertson and Sparck Jones advise to assign $b = 0.75$, which in our view reflects the situation, that most of the long e-mails do not cover multiple topics. The parameter k can be used to manipulate the effect of the term frequency on the relevance weight. Assigning $k = 0$ reflects the assumption that term frequency does not have any influence on the relevance of a certain term in a document. Robertson and Sparck Jones advise to assign $k = 2$, which is a value that reflects the influence of the term frequency in the TREC reference test set.

Profile based classification

In section 4.3.1 we presented an explanation of the Rocchio classification approach. This classification approach uses positive and negative examples to build a centroid representation for each category. The parameters α and β manipulate the influence of the positive examples and the negative examples respectively. Moschitti (2003) describes a method to determine the optimal values for the parameters α and β by relating both parameters using the parameter ρ. The parameter ρ is defined by β/α and the optimal parameter setting for the Rocchio classifier can be determined by increasing ρ smoothly until an optimum in classification accuracy is reached. The nature of the Rocchio classifier is to determine an optimal margin between the centroids of categories. This is similar to Support Vector Machines, but these do not determine a margin between centroids, but between the complete data collections (with a minimal classification error). This optimal margin is based on the occurrence of index terms in the centroid. The parameter estimation works as a feature selection procedure, and selects only those features that are relevant to distinguish a category from the other categories. By increasing the influence of the negative examples with respect to the positive examples, we smoothly remove terms from the positive examples that are irrelevant for distinguishing this category from the others. This process continues until the influence of the negative examples has become that big that relevant terms are being removed from the centroid (and the accuracy as a function of ρ has reached its maximum).

Figure 8.2 shows the classification accuracy as a function of ρ for a random test set of our e-mail corpus, using the Profile Based/Okapi/Inner product classification combination. After the classification accuracy has reached its maximum, the relevant and weak relevant terms will be eliminated, so this is the optimal value for ρ and the optimal ratio between α and β.

A value for $\rho \leq 1$ indicates that the positive examples are more important than the negative examples in the classification process, which intuitively makes sense. Joachims (1997) describes that the default values for α and β are 16 and 4 respectively ($\rho = 0.25$), values that are extensively used in text classification research (Buckley et al., 1994). Moschitti (2003) shows in his research on parameter estimation for the Rocchio classifier, that the best classification accuracy is reached when $\rho > 1$, which means that the negative examples have a greater influence on classification accuracy then the positive ones (on the Reuters corpus).

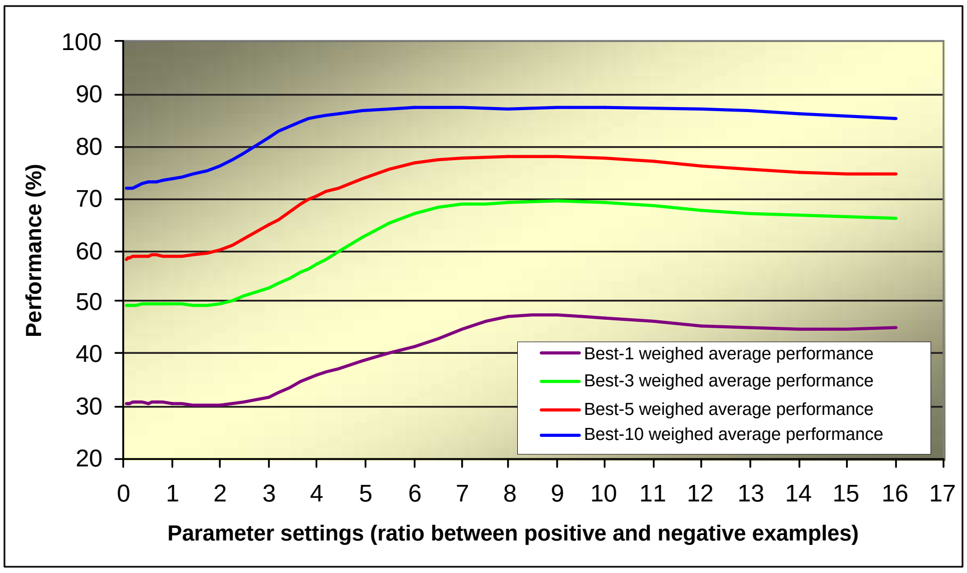

Figure 8.2: Parameter estimation for $\rho = \beta/\alpha$, for the profile based classification method. The optimal parameter setting $\rho = 8$ reaches a best-5 classification accuracy over 78%.

Figure 8.2 shows the classification accuracy as a function of ρ for our classification problem and confirms the conclusion of Moschitti. The Parametrized Rocchio Classifier (as Moschitti calls it) outperforms the Rocchio classifier with optimal parameter settings discussed in literature (in which $\rho = 1$) with 5 percentage points and even 10 percentage points when a more general parametrization is used (e.g. $\rho = 0.25$).

In our e-mail classification problem the optimal parameter setting is: $\alpha = 1$ and $\beta = 8$ (and thus $\rho = 8$), meaning that the influence of the negative examples is 8 times as big as the positive examples for the best classification results. By setting $\rho = 8$ we yield a best-5 classification accuracy of 78.15%, opposed to the accuracy of 58.85% if we use the default parameter settings suggested by Buckley et al. (an increase of almost 20 percentage points!).

Example based classification

The K-nearest-neighbour classification approach bases the classification on the K most relevant documents (neighbours) retrieved by an information retrieval system. One might argue that we only need to consider the best retrieved document, since the information retrieval routine determined that this document is most relevant and might therefore require the same answer. However, the first retrieved document (that is syntactically most similar to the query document) is not necessarily the document that is semantically most similar to the query document. For example, the second or even ninth document might reflect the meaning of the new document better than the first document. This means that the words in the e-mail may overlap for great amount, but the actual question posed in both mails differs. $K = 1$ would therefore probably be not the best approach to optimize the classification accuracy, but a value for K that is too large would neither. In the latter case, the largest categories would finally get the upper hand in classification accuracy, because these categories would be suggested very often because they simply contain more messages than the others. The value for K has to be empirically determined for every text classification problem. Typical values for K in text classification problems range over $K \in \{10, ..., 100\}$. Initial tests on our e-mail corpus determined that assigning $K = 50$ yields the best classification results, and is therefore used in the experiments. The differences in classification accuracy between values for $K \in \{40, 50, 60\}$ are small (a difference of approximately 1 percentage point), and the optimum value for *K* might change if we apply different term weighting schemes or apply language technology. However, it is important to use the same value for *K* in every experiment, otherwise we would not be able to determine the influence of a certain language technology on the classification accuracy.

8.2 Baseline classification experiments

To prove our first hypothesis that IR based classification performs better than the current classification approach used by Em@ilco, the first experiments are fully focussed on the classification methods. For the experiment series in this section we compare the results of the combinations of two classification methods: Profile based classification (section 4.3.1) and Example based classification (section 4.3.2), two term weighting schemes: Okapi term weighting (section 3.3.2) and TF.IDF term weighting (section 3.3.1) and two similarity measures: cosine and inner product.

We have measured the best-*n* performance for each of the categories, and calculated the average performance. The average performance is calculated by averaging the performance per category, and does not take the size of categories into account. We also measured the **total** number of e-mails that is correctly classified within the top-*n* results of the system. This performance is called the weighted average performance, since it also takes the category sizes into account. The weighted average performance of the system is calculated by dividing the total number of correctly classified mails within the top-*n*, by the total number of incoming mails. The weighted average performance best reflects the usability of such a classification system in a contact centre, as it represents the total number of mails that is correctly classified by the system.

8.2.1 Baseline experiment

In our baseline experiments we calculated the best-*n* performance for each category, and for each $n \in \{1..37\}$ (37 being the maximum number of categories). The average of these performances can be found in the result tables in the column *Avg*. The weighted best-*n* performance of the classification system are listed in

Approach			% Best-1 Avg	% Best-1 Wavg	% Best-3 Avg	% Best-3 Wavg	% Best-5 Avg	% Best-5 Wavg
Example	Okapi	IP	36.67	**52.07**	59.45	74.63	69.50	82.15
Example	Okapi	Cos	39.04	48.20	61.15	**77.15**	72.28	**84.82**
Example	TFIDF	IP	9.96	25.80	19.95	43.11	33.43	56.75
Example	TFIDF	Cos	37.94	47.25	60.54	76.35	70.90	83.90
Profile	Okapi	IP	**43.9**4	45.87	**67.23**	68.44	**77.78**	77.40
Profile	Okapi	Cos	39.31	27.49	64.49	52.39	75.84	64.09
Profile	TFIDF	IP	41.25	43.07	65.80	67.11	76.52	76.50
Profile	TFIDF	Cos	39.27	27.20	62.93	50.61	74.43	63.08
Em@ilco approach			14.78	14.21	29.37	31.94	37.18	40.00

Table 8.1: Classification results of the baseline experiment. The table lists the average (Avg) and weighted average (Wavg) best-n performance for all of the classification approaches for $n \in \{1, 3, 5\}$.

the columns *Wavg* in the result tables.

Table 8.1 displays the results of the baseline test in which we experiment with the eight possible combinations of classification method, term weighting scheme and similarity measure. The first column of the approach section in the table lists the used method (Profile based or Example based), the second column lists the term weighting scheme (Okapi or TF.IDF) and the third column lists the similarity measure (*IP* for inner product and *Cos* for cosine) we used. The results listed in this table only cover the best-n performance for $n \in \{1, 3, 5\}$. The results for $n = 1$ show the classification accuracy (and thus answer accuracy) if e-mails in this contact centre were automatically answered. For each experiment, the best average and best weighted average are printed in bold. The average and weighted average performances of the Em@ilco approach are printed on the last row (these performances have been measured using the final test set discussed in section 8.1.1). Figure 8.3 shows the differences between the best example based method and the best profile based methods, in terms of average and weighted average performance. The performance of the example based methods is printed in blue, the performance of the profile based methods in red and the performance of the Em@ilco approach is printed in black, while the average performance is marked using a triangle and the weighted average performance using a square. The green line denotes the best-guess performance (only weighted average) and illustrates the percentage of correct answer suggestions if we suggest the best-*n* answers in descending order of category size (i.e. the largest category (answer suggestion) is the best suggestion, the second largest category the second best, etc.).

Analysis and discussion

First of all, we can consider our first hypothesis proven: information retrieval based classification methods outperform the Em@ilco classification method in terms of classification accuracy for our best classification approaches.

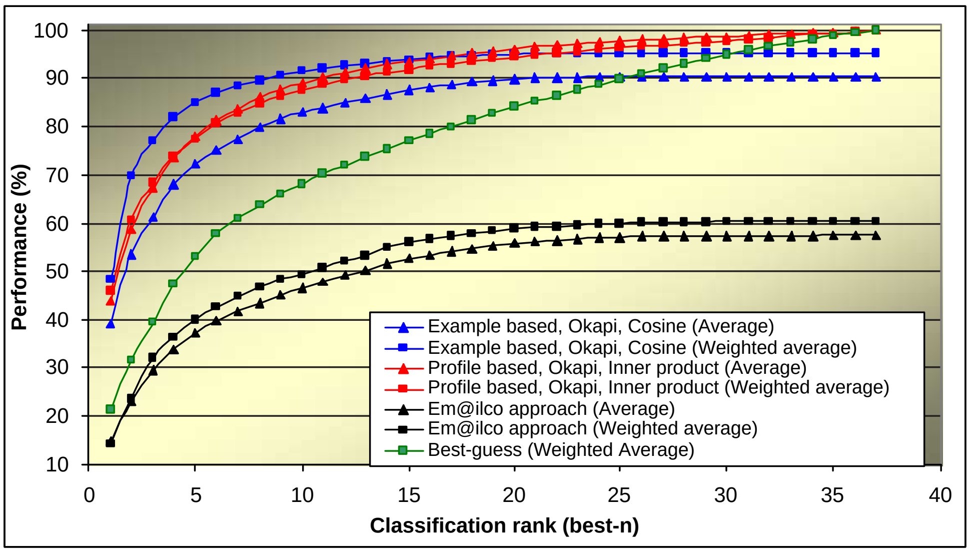

Figure 8.3: Performance of the best example based method and the best profile based method, in terms of average and weighted average performance, compared to the Em@ilco approach

In table 8.1 we printed the maximum classification accuracy per best-n average and weighted average performance measure in bold. The profile based classification methods reach a higher average classification accuracy than the example based methods. However, the example based classification methods yield a higher classification accuracy in terms of best-n weighted average performance. This difference between average and weighted average performance is also illustrated in figure 8.3. In this figure we see that the average and weighted average performance of the profile based classifier (denoted with red lines) are approximately equal. But the difference in average and weighted average performance of the example based classification method is 10 percentage points for the best-5 performance. This difference is caused by the high classification accuracy of the example based method in the large categories (see figure 8.4). The difference between weighted classification accuracy of the profile and example based methods can be explained using figure 8.4. The profile based methods (denoted with the red bars) perform really good on the smaller categories (100 to 350 e-mails per category). However, the example based methods (denoted with the blue bars) perform really good on the larger categories (400 to 3590 e-mails per category), which results in a higher total (weighted) classification accuracy because it simply classifies more e-mails correctly than the profile based method. Recall from section 4.3.1 that the weakness of the Rocchio classifier (profile based method) is that it determines just one centroid per category. If a category consists of many mails, it is more difficult to determine a single centroid that very accurately reflects all the mails in the category. The mails in the category all require the same answer, and therefore pose similar questions, but the more e-mails that are stored in a category, the more different ways of posing the question are en-

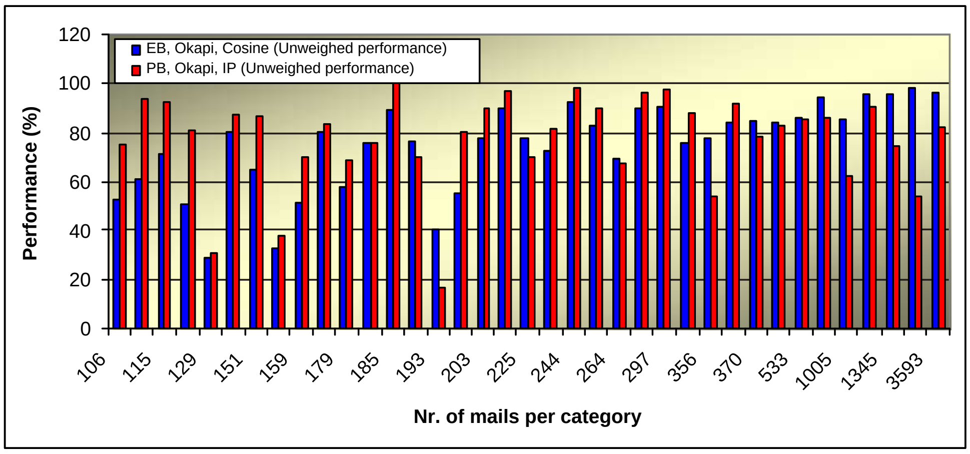

Figure 8.4: Unweighted performance of the EB/Okapi/Cosine and the PB/Okapi/IP approach, with respect to the category size

countered, resulting in a great variety of words.

Remarkably, the EB/TFIDF/IP combination performs really poor: a 26.71 percentage point difference with the EB/Okapi/IP combination in average best-1 performance (more than 3.5 times as bad), and more than two times as bad when measuring weighted best-1 performance. This difference is caused by the lack of document length normalization in the TF.IDF weighting scheme and inner product similarity measure combination (recall that the cosine similarity measure and the Okapi weighting scheme both have built in length normalization). For the other example based combinations (in which document length normalization is applied), the performance differences are not that large. According to the results of table 8.1, this explanation does not hold for the profile based methods. For the profile based methods, the cosine similarity measure doe not seem to work well (for both Okapi and TF.IDF weights). Hiemstra (2000) also compared the TF.IDF weighting scheme using the cosine similarity (called the tfc.tfc method) measure with the Okapi weighting scheme using the inner product (called the BM25 formula) and found that the Okapi and inner product combination performed about 100% better than the TF.IDF cosine combination in average precision. Overall we can say that the example based combination using the Okapi weighting scheme and cosine similarity measure performs best for this e-mail classification problem.

The fact that the best-guess performance is higher than the performance of the Em@ilco approach illustrates two things. First of all, a great amount of the mails is about a relative small set of the questions, resulting in a couple of large categories, which account for a large amount of the answers. If the categories were of equal size, the best-guess performance would be represented by a straight line from 0 to 100%, and perform worse than the Em@ilco approach until the best-15 to best-20 answer suggestions. Moreover, this difference illustrates that it is not that trivial to manually determine sets of relevant and distinctive keywords for each standard question.

8.3 Language technology experiments (1)

In our second hypothesis, we stated that language technology improves the classification results of information retrieval based classification methods. Because of the unstructured and easily accessible status of e-mail as a means of communication, spelling errors and grammatically inconsistencies may occur. To prove this hypothesis, we continue experimenting with the e-mail classification system, but pre-process the e-mail messages using language technology discussed in chapter 6, in both the training and the classification process. First, we will illustrate the influence of each individual technology on classification accuracy. For the next series of experiments, the most promising techniques will be combined.

To illustrate the influence of each individual language technology on the classification results, we show the increase or decrease in percentage points, with respect to the classification accuracy of the baseline experiments of section 8.2.1. An experiment is conducted for each example and profile based combination of the baseline test, of which the results are displayed in the tables for example and profile based methods in this section. In both tables we present the change in classification accuracy due to the application of a language technology for best-*n* performance in weighted average. A decrease in accuracy is marked red and in each table the used language technologies are abbreviated according to the next list:

Abbreviation	**Language technology**
sw	Stopword removal
dc	Decompounding
ng	N-gram check
st	Stemming
sp	Spelling correction
wc	Word classes (Part of speech tagging)

8.3.1 Example based experiments

This section discusses the the experiments we conducted using language technology in combination with the example based classification methods. In table 8.2 we present the differences in classification accuracy for the language technology experiments with respect to the baseline experiments of section 8.2.1. The classification accuracies for each language technology experiment are also listed in appendix A.1. In the table and the discussion of the results we will abbreviate the example based method to EB, the inner product to IP and the cosine to Cos.

Experiment		Weighted average performance (%)						
n	**Approach**	**base**	**sw**	**dc**	**ng**	**st**	**sp**	**wc**
1	Okapi/IP	52.07	+3.58	+6.48	+5.87	+5.81	-11.19	+6.24
	Okapi/Cos	48.20	-7.76	-0.26	+2.08	+8.73	+10.15	+10.05
	TFIDF/IP	25.80	+14.97	+0.89	+0.03	-0.33	-0.02	+14.07
	TFIDF/Cos	47.25	-6.56	-0.32	-11.13	+6.61	+7.43	+10.56
3	Okapi/IP	74.63	+3.10	+6.42	+6.10	+5.96	-12.75	+6.01
	Okapi/Cos	77.15	-3.69	+0.35	+0.95	+0.91	+2.09	+1.46
	TFIDF/IP	43.11	+20.12	+0.85	+0.19	-0.95	+0.14	+17.55
	TFIDF/Cos	76.35	-2.59	+0.20	-10.70	+0.17	+0.96	+1.98
5	Okapi/IP	82.15	+2.63	+5.25	+5.22	+4.74	-10.63	+4.69
	Okapi/Cos	84.82	-3.15	+0.04	+0.33	+0.53	+1.26	+0.45
	TFIDF/IP	56.75	+17.79	+0.37	+0.17	-1.52	+0.11	+16.01
	TFIDF/Cos	83.89	-2.02	+0.07	-7.58	+0.29	+0.74	+1.41
10	Okapi/IP	89.47	+2.00	+3.49	+3.45	+3.38	-6.03	+3.15
	Okapi/Cos	91.42	-2.05	+0.17	+0.48	+0.27	+0.63	+0.25
	TFIDF/IP	71.78	+13.39	+0.43	+0.23	-0.15	+0.55	+12.25
	TFIDF/Cos	91.02	-1.60	+0.28	-5.64	+0.16	+0.46	+0.64

Table 8.2: Influence of the use of language technology for the **Example based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

Stopword removal

Stopword removal causes a slight improvement in classification accuracy for the Okapi/IP combination and a huge improvement for the TFIDF/IP combination (13 to 20 percentage points increase). The improvement for the TFIDF/IP combination may be explained by the lack of document length normalization of this method. If all stopwords are removed from the documents, differences in document length become much smaller. If we apply document length normalization using the cosine similarity function, the classification accuracy decreases with 2 to 2.5 percentage points (and 6.5 to 7.7 for best-1 performance). Stopword removal seems to work best if the inner product similarity measure is used, because no strong document length normalization is applied. The document length normalization of the Okapi weighting scheme is related to the average document length, and not the absolute length of a certain document, which makes it less strong then the cosine document length normalization.

The huge increase in classification accuracy for the TFIDF/IP combination does not cause the total classification accuracy of this method to exceed the 84.77% accuracy of the baseline Okapi/Cosine combination (see appendix A.1).

Decompounding

Decompounding (or compound splitting) only increases classification accuracy for the Okapi/IP combination, with an increase in weighted average performance ranging from 3.49 to 6.48 percentage points. Because of this increase in accuracy, this combination now outperforms the EB/Okapi/Cosine combination by an increase of approximately 2.5 percentage points:

Combination	**Accuracy**
EB/Okapi/IP with dc	87.40%
EB/Okapi/Cos without dc	84.82%
EB/Okapi/Cos with dc	84.87%

N-gram check

The N-gram check improves the classification accuracy for the Okapi/IP combination (3.45 to 6.10 percentage points), while it worsens the accuracy of the TFIDF/Cosine combination (a decrease ranging from 11.13 to 5.64 percentage points). The increase in accuracy of the Okapi/IP combination (with 87.36% in best-5 weighted average performance) causes this method to perform better than the baseline performance of the Okapi/Cosine method, but still not better than the Okapi/IP combination using decompounding (respectively 84.82% and 87.40% in best-5 weighted average performance).

Stemming

Stemming does not significantly influence the classification accuracy of the Okapi/Cos combination and TF.IDF based combinations. However, applying stemming results in a considerable increase of best-1 performance. Also, the Okapi/IP combination benefits from using stemming with an increase in weighted average performance of 3.38 to 6.32 percentage points and a total classification accuracy of 86.88% for best-5 performance (exceeding the results of our baseline experiment). Since the performance for $n = 1$ increases and for $n \in \{3, 5, 10\}$ remains approximately zero (except for the Okapi/IP combination) we can conclude that stemming mainly influences the ranking of the retrieved documents instead of improving overall classification accuracy. Again, document length normalization comes into play: if strong document lenght normalization is applied (using cosine), stemming does not siginificantly influence the classification results. Also, when the documents are not normalized (TFIDF/IP combination), the influence of stemming is also negligible. The difference in classification accuracy using stemming is caused by the difference between the Okapi and TF.IDF term weighting scheme, of which the first is more sophisticated.

Spelling correction

Spelling correction slightly improves classification accuracy of the Okapi/Cos and TFIDF/Cos combinations, does not significantly influence the TF.IDF/IP combination, but dramatically decreases the accuracy of the Okapi/IP combination. Spelling correction causes a decrease of 12.75 percentage points for best-3 weighted average performance and 10.63 percentage points for best-5 weighted average performance. Most likely, spelling correction introduces more noise in the form of wrongly corrected words, than it can take away by correcting mis-

Experiment		**Weighted average performance (%)**						
n	**Approach**	**base**	**sw**	**dc**	**ng**	**st**	**sp**	**wc**
1	Okapi/IP	45.87	-1.76	-4.11	-0.26	+0.37	-0.07	+1.16
	Okapi/Cos	27.49	+0.47	-1.76	+0.30	-1.42	-0.17	+1.04
	TFIDF/IP	43.07	-0.62	-6.20	-0.50	+0.10	-0.06	+2.33
	TFIDF/Cos	27.20	+0.02	-2.06	+0.29	-1.71	-0.17	+0.58
3	Okapi/IP	68.45	-0.78	-3.72	-0.27	-0.24	-0.10	+1.06
	Okapi/Cos	52.39	+2.14	-3.61	+0.16	-0.73	-0.20	+2.13
	TFIDF/IP	67.11	+0.01	-4.47	-0.21	+0.06	+0.09	+1.73
	TFIDF/Cos	50.62	+2.14	-3.91	+0.37	-0.86	-0.01	+2.18
5	Okapi/IP	77.40	-0.02	-2.70	-0.14	-0.56	-0.16	+0.91
	Okapi/Cos	64.09	+2.30	-3.17	+0.10	+4.19	-0.25	+1.98
	TFIDF/IP	76.50	+0.28	-3.10	+0.05	-0.30	-0.05	+1.30
	TFIDF/Cos	63.08	+2.66	-3.79	-0.01	-0.17	-0.13	+1.85
10	Okapi/IP	87.51	+0.40	-1.69	-0.04	-0.45	-0.17	+0.57
	Okapi/Cos	80.53	+1.89	-2.76	-0.21	-0.29	-0.37	+1.16
	TFIDF/IP	87.21	+0.48	-1.64	+0.16	-0.40	-0.12	+0.68
	TFIDF/Cos	80.02	+2.09	-2.62	-0.04	-0.08	-0.29	+1.46

Table 8.3: Influence of the use of language technology for the **Profile based** methods. The first column lists the value for n to measure the best-n performance in weighted averages

spelled words. Apparently, TFIDF is relatively insensitive to this noise, while Okapi is not.

Part of speech tagging

The use of only nouns, verbs and adjectives for classification improves the accuracy of all example based classification methods. However, the increase in classification accuracy is much bigger for classification methods using the inner product similarity measure, than for classification methods using the cosine similarity measure. Again, this difference is caused by the strong document length normalization of the cosine function. The huge increase of accuracy for the TFIDF/IP combination causes the total classification accuracy of this combination to become 72.76% (see appendix A.1), which is still 12 percentage points lower than the baseline performance of our best classification approach.

8.3.2 Profile based experiments

This section discusses the experiments we conducted using language technology for the profile based classification methods. In table 8.3 we present the differences in classification accuracy for the language technology experiments with respect to the baseline experiment of section 8.2.1. In appendix A.1 the classification accuracies are also listed for each language technology experiment. In the table and the discussion of the results we will abbreviate the profile based method to PB.

Applying language technology to e-mails before they are sent to the classification process does not cause a significant increase in classification accuracy of the profile based methods, for most language technologies. Eliminating stopwords slightly increases the accuracy for the cosine based classification methods, but does not significantly influence the accuracy of the inner product based methods. Applying the N-gram check, stemming or spelling correction also does not significantly influence the accuracy. If we split compound terms, the classification accuracy decreases with 2 to 6 percentage points, and if we only use nouns, verbs and adjectives in classification, the accuracy increases with 1 to 2 percentage points.

Remarkably, the classification accuracy is only influenced by language technologies that introduce or eliminate words. Eliminating stopwords or eliminating words that are not nouns, verbs or adjectives causes a slight increase in classification accuracy, while the introduction of words (which is the result of compound splitting) causes a decrease in accuracy. This can be explained by the choice of the specific parameter settings for the Rocchio classifier. The parameter settings are optimal for the Okapi and inner product combination for unchanged e-mails (i.e. no language technology is applied). This means that the other classification approaches (based on the cosine similarity measure) may not be working with the optimal parameter settings and therefore might be improved by applying language technology. The process of parameter selection is an optimalization of feature selection: if we increase ρ we eliminate more features (words) from the centroid representation until a certain maximum is reached. After this maximum, relevant and weak relevant terms will also be eliminated (which does not improve the classification process). Since the optimal parameter settings for the Okapi/IP classification method have been determined on the set of unchanged e-mails, the classification accuracy for this combination will most likely not be improved by applying language technology. However, the other classification combinations might be improved by language technology, if that language technology compensates for the not optimal feature selection process (that is done by parameter estimation for ρ). If the language technology eliminates irrelevant features from the e-mails, it compensates for the not optimal parameter settings for the Rocchio classification algorithm. This is also the reason that eliminating stopwords (irrelevant words) improves classification accuracy for the cosine based methods. Also, only using nouns, verbs and adjectives (accounting for almost all relvant and successful query terms of a document, according to Kraaij and Pohlmann, 1996) eliminates most irrelevant terms from a document. By applying compound splitting, two or more words (parts of the compound) are introduced in the documents and only one is eliminated for each compound term. This process does not assist in feature selection as it does not eliminate irrelevant features from the document, but only introduces (both relevant and irrelevant) terms. However, if we choose an optimal parameter setting for each specific classification method, the application of language technology on these classification methods would probably not have a signinficant influence on the classification accuracy.

8.4 Language technology experiments (2)

The application of individual language technologies to the profile based classification methods did not result in a significant increase in classification accuracy. Therefore, applying combinations of these language technologies would probably not positively influence the classification accuracy either. For the example based classification methods, the application of language technology does positively influence the classification accuracy for most classification approaches. Spelling correction is an exception on this statement: it does not significantly improve the accuracy (for best-n performance for $n \in \{3, 5, 10\}$) and is therefore excluded from the following experiments. In this section we focus on applying combinations of language technology to the example based classification approaches, in order to improve the total classification accuracy of these methods. Unfortunately, it is not feasible to conduct all permutations of language technology combinations, which is $6! = 720$ experiments, since the order of applying different technologies also influences the classification. For this reason, we start experimenting with binary combinations of stopword removal, decompounding, N-gram checking, stemming and pos-tagging (using only nouns, verbs and adjectives). If the results of these binary combinations are promising, we conduct further experiments with the combination of language technologies. To demonstrate the lack of positive influence of applying language technology for the profile based classification methods, we also conduct these experiments with the profile based classifier.

8.4.1 Example based experiments

In this section we present the results of the experiments using the example based classification methods, in which we combine the language technologies presented in chapter 5. Table 8.4 displays the influence of the binary combinations of language technology for the example based classification methods.

Word classes and Stemming

In this combination we only use nouns, verbs and adjectives, of which the nouns and verbs are also stemmed. This language technology combination yields no significant improvement for the EB/IP combinations with respect to the influence of only using word classes (respectively 8 and 17 percentage points decrease in the Okapi/IP and the TFIDF/IP combination). With respect to the application of just stemming, this combination results in a considerable decrease in accuracy for the Okapi/IP combination and performs slightly better for the TFIDF/IP combination. For both the Okapi/Cos and TFIDF/Cos approaches, this combination has no significant influence on the classification accuracy.

Since combining word classes and stemming does not improve the classification accuracy, we will not further investigate other combinations based on this approach. Using either stemming or word classes would cause a higher increase in classification accuracy than the combination of both.

Experiment		**Weighted average performance (%)**						
n	**Approach**	**wc/st**	**sw/dc**	**sw/ng**	**dc/st**	**sw/st**	**dc/wc**	**ng/st**
1	Okapi/IP	-5.50	+3.99	+4.39	+5.19	+7.16	+5.92	+6.11
	Okapi/Cos	+9.57	+6.63	-0.24	+7.57	+10.09	+9.61	+3.73
	TFIDF/IP	+4.99	+15.67	+15.53	-0.05	+17.06	+14.43	-0.31
	TFIDF/Cos	+10.08	+6.87	+0.25	+5.81	+10.58	+10.35	+3.47
3	Okapi/IP	-10.36	+4.71	+4.43	+6.24	+6.84	+6.11	+6.27
	Okapi/Cos	+0.89	-0.55	-1.78	+0.67	+2.22	+1.56	+0.66
	TFIDF/IP	+3.31	+19.54	+21.15	+0.04	+24.05	+16.57	-0.99
	TFIDF/Cos	+1.65	+0.16	-1.09	-0.05	+3.02	+2.01	+0.18
5	Okapi/IP	-12.79	+3.78	+3.73	+4.92	+6.19	+4.78	+4.81
	Okapi/Cos	+0.22	-0.78	-1.99	+0.31	+1.67	+0.50	+0.07
	TFIDF/IP	-1.05	+17.25	+18.52	+0.17	+21.59	+15.19	-1.65
	TFIDF/Cos	+1.02	+0.08	-1.18	+0.23	+2.79	+1.47	+0.22
10	Okapi/IP	-15.27	+2.60	+2.71	+3.45	+3.87	+3.25	+3.42
	Okapi/Cos	+0.17	-0.71	-1.63	+0.22	+1.10	+0.12	-0.03
	TFIDF/IP	-6.35	+13.38	+13.95	+0.71	+16.12	+11.60	-0.16
	TFIDF/Cos	+0.72	-0.29	-1.08	+0.19	+1.63	+0.46	+0.07

Table 8.4: Influence of the use of binary combinations of language technology for the **Example based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

Stopword removal and decompounding

If we first remove all stopwords from the document, and then split compounds, the classification accuracy increases with respect to only removing stopwords. This increase ranges from 2 (best-5 performance) to 13 (best-1 performance) percentage points for the cosine based approaches and approximately 1 percentage point for the Okapi/IP combination. For the Okapi/Cosine combination, the classification accuracy is comparable to that of only removing stopwords.

With respect to decompounding, the combination of removing stopwords and splitting compounds has no significant influence.

Stopword removal and N-gram check

Applying the N-gram check after the stopwords have been removed results in a small increase in classification accuracy with respect to only stopword removal, for all classification approaches (about 1 percentage point).

With respect to the classification accuracy of only applying the N-gram check, we see an increase of 6 to 18 percentage points for the TFIDF combinations (TFIDF/Cos and TFIDF/IP respectively) and a decrease of approximately 2 percentage points for the Okapi combinations.

Decompounding and stemming

Splitting the compounds and stemming all the remaining words of a document (including the resulting words of the decompounding process) does not have a significant influence on the classification results with respect to using only one of these techniques.

Stopword removal and stemming

The combination of stopword removal and stemming appears to be a good language technology application for all of our example based classification approaches. The classification accuracy increases with at least 4 percentage points with respect to only applying stopword removal.

With respect to the classification accuracy yielded by applying stemming for the example based classification approaches, this combination shows an increase of 1.5 a 2 percentage points for all our classification approaches, except for the TFIDF/IP combination. For this approach, the increase in in classification accuracy reaches a maximum of 21 percentage points.

Wordclasses and decompounding

The combination of using decompounding and only using nouns, verbs and adjectives for the classification process does not have a significant influence on the classification accuracy if we compare it to only applying decompounding or only using nouns, verbs and adjectives for classification. An exception can be notices for the Cosine combinations in best-1 performance, which show a considerable increase in accuracy. However, the other performance measures do not reflect this increase.

N-gram check and stemming

With respect to applying the N-gram check to the words of the documents, we do not yield a significant increase in classification accuracy for the Okapi based methods, and the TFIDF/IP method. However, for the TFIDF/Cosine method, the classification accuracy increases with 5 to 14 percentage points (for best-10 and best-1 performance respectively).

Applying the combination of the N-gram check and stemming only significantly influences the best-1 performances of the Okapi/Cos and the TFIDF/IP combination. For best-n performance with $n \in \{3, 5, 10\}$, the classification accuracy remains approximately constant with respect to the application of only applying stemming.

Experiment		Weighted average performance (%)						
n	Approach	wc/st	sw/dc	sw/ng	dc/st	sw/st	dc/wc	ng/st
1	Okapi/IP	-2.27	-5.95	-2.55	-3.40	-3.36	-3.35	-0.04
	Okapi/Cos	-0.56	-1.51	+0.56	-3.64	-1.25	-1.04	-0.97
	TFIDF/IP	-1.45	-7.02	-1.19	-4.59	-1.39	-4.67	-0.24
	TFIDF/Cos	-0.60	-2.53	+0.39	-3.38	-1.20	-1.57	-1.17
3	Okapi/IP	-1.85	-5.01	-1.21	-3.41	-0.66	-2.33	-0.44
	Okapi/Cos	+2.12	-2.24	+1.85	-4.79	-2.71	-1.58	-0.02
	TFIDF/IP	-1.17	-5.02	-0.65	-3.97	+0.54	-2.70	-0.07
	TFIDF/Cos	+1.98	-2.92	+2.04	-5.37	-2.53	-1.89	+0.00
5	Okapi/IP	-1.56	-3.87	-0.48	-3.34	-0.05	-1.58	-0.25
	Okapi/Cos	+2.74	-1.86	+2.04	-3.65	-2.23	-0.96	+0.44
	TFIDF/IP	-0.89	-3.11	-0.05	-3.22	+0.85	-1.41	+0.10
	TFIDF/Cos	+2.94	-1.53	+2.35	-4.10	-1.78	-1.57	+0.71
10	Okapi/IP	-1.11	-1.93	+0.37	-2.43	+0.11	-1.21	-0.34
	Okapi/Cos	+1.52	-1.79	+1.71	-2.86	-1.97	-0.99	+0.26
	TFIDF/IP	-0.82	-1.45	+0.67	-2.12	+0.52	-1.06	+0.04
	TFIDF/Cos	+1.99	-0.76	+1.99	-2.63	-1.10	-0.85	+0.81

Table 8.5: Influence of the use of binary combinations of language technology for the **Profile based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

8.4.2 Profile based experiments

As we expected in the introduction of this section, the application of combinations of language technologies does not improve the classification accuracy for the profile based classification methods. In the last section we already discussed that the application of language technology only might improve the classification accuracy if it compensates for the sub-optimal feature selection that is implied by the parameter settings for the Rocchio classifier. For almost all binary combinations we experimented with, the classification accuracy equals, or is worse then, the application of only one language technology from the combination. However, the combination of stopword elimination and N-gram check slightly increases the classification accuracy with respect to only eliminating stopwords or only applying the N-gram check. Apparently, stopword elimination and N-gram checking both compensate the non-optimal parameter settings for feature selection of the Rocchio classifier. Again, the increase is only achieved in the classification methods for wich the parameter settings are not optimal.

8.5 Language technology experiments (3)

In the last section we experimented with binary combinations of language technology for our classification methods. In those experiments we noticed that the combination of two individually successful language technologies do not always result in a better classification accuracy than the individual ones. For example, stemming and the use of only nouns, verbs and adjectives both individually cause

Experiment		Weighted average performance (%)			
		Example based		Profile based	
n	**Approach**	**sw/dc/st**	**sw/ng/st**	**sw/dc/st**	**sw/ng/st**
1	Okapi/IP	+4.05	-10.66	-7.18	-4.18
	Okapi/Cos	+6.17	-0.68	-3.27	-0.96
	TFIDF/IP	+15.45	+14.70	-6.08	-2.35
	TFIDF/Cos	+7.61	+0.62	-3.58	-1.16
3	Okapi/IP	+4.25	-12.57	-5.06	-1.07
	Okapi/Cos	-0.97	-2.13	-4.07	+1.33
	TFIDF/IP	+18.81	+17.93	-3.98	+0.20
	TFIDF/Cos	+0.01	-1.30	-4.18	+1.42
5	Okapi/IP	+3.69	-14.20	-3.86	+0.14
	Okapi/Cos	-1.04	-2.02	-2.37	+2.20
	TFIDF/IP	+16.67	+10.50	-2.60	+1.02
	TFIDF/Cos	+0.07	-1.23	-1.88	+2.83
10	Okapi/IP	+15.45	+14.70	-6.08	-2.35
	Okapi/Cos	-0.65	-1.63	-2.25	+1.64
	TFIDF/IP	+12.83	+1.12	-1.50	+0.63
	TFIDF/Cos	-0.04	-0.98	-1.04	+2.35

Table 8.6: Influence of the use of combinations of three language technologies for the **Profile based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

an increase in classification accuracy of about 5 percentage points with respect to the baseline experiment for best-5 performance of the EB/Okapi/IP combination. However, the combination of these two technologies (st/wc) causes the classification accuracy to decrease with 12.79 percentage points!

The following language technology combinations performed better than their individual technologies, or at least perform as well as them. These are the combinations on which we want to base our next series of experiments.

Stopword removal	/	decompounding
Stopword removal	/	N-gram check
Stopword removal	/	Stemming
decompounding	/	stemming
N-gram check	/	stemming

Combining these technologies (in combinations of three distinct technologies) results in the combination of stopword removal, decompounding, stemming (sw/dc/st) and stopword removal, n-gram check and stemming (sw/ng/st). The results of these experiments for the example and profile based classification methods are presented in table 8.6.

As illustrated in table 8.6, the profile based classification methods do not show a significant increase in classification accuracy for the combination of stopword elimination, decompounding and stemming. The slight increase in classification accuracy for some of the methods in which the language technology combination of eliminating stopwords, applying the N-gram check and stemming is applied, does not exceed the increase in accuracy of the binary combination of applying the N-gram check eliminating stopwords (see table 8.5).

Applying the combination of eliminating stopwords, decompounding and stemming to the example based classification approaches does cause an increase in classification accuracy with respect to the baseline experiments. However, this increase in accuracy is smaller than the increase caused by applying only the combination of eliminating stopwords and decompounding. The use of stemming for this combination does not cause an extra increase in classification accuracy for the stopword removal and decompounding combination. The combination of removing stopwords, applying the N-gram check and stemming for the example based classification approaches does not yield an increase in accuracy for the Okapi based methods. The increase in classification accuracy for the TFIDF combinations does not exceed the classification accuracy for combination of only applying the stopword removal and N-gram check combination.

Since no extra increase in classification accuracy is yielded from combining three language technologies to the classification process, we see no reason to further investigate the influence of other combinations of language technology.

8.6 Conclusion

First of all, IR based classification approaches perform better in suggesting relevant answers to e-mail than the keyword approach implemented by Em@ilco. The example based methods (K-NN classifier) outperform the profile based methods (Rocchio classifier) in terms of weighted average classification accuracy. For an optimal use of the classification algorithms presented in this thesis, the optimal parameter settings have to be empirically determined. In case of the example based classifier the value for K (which is the number of relevant documents that influence the classification results) for the K-Nearest-Neighbour algorithm has to be determined. In case of the profile based classifier, the optimal value for $\rho = \beta/\alpha$ has to be determined. This ratio ρ manipulates the influence of the negative training examples with respect to the influence of the positive training examples. For every classification problem these values may differ from the values in this thesis.

The second hypothesis, which states that language technology improves the classification accuracy for the IR based classification approaches, cannot be answered with an unambiguous yes. The application of language technology for e-mail classification using the example based classifier indeed improves the classification accuracy. Figure 8.5 illustrates this statement. In this figure we present the classification accuracies of our best example and profile based classification method (Okapi weights and inner product similarity) and the best application of language technology to both approaches. The example based classification approaches are denoted by the blue lines and the profile based classification approaches are denoted by the red lines. The baseline classification accuracy is marked using squares, while the classification accuracy when we also use language technology are marked using triangles. For the example based classifier, the application of decompounding yields the best classification accuracy and shows an increase in best-5 accuracy of 5.25 percentage points with respect to

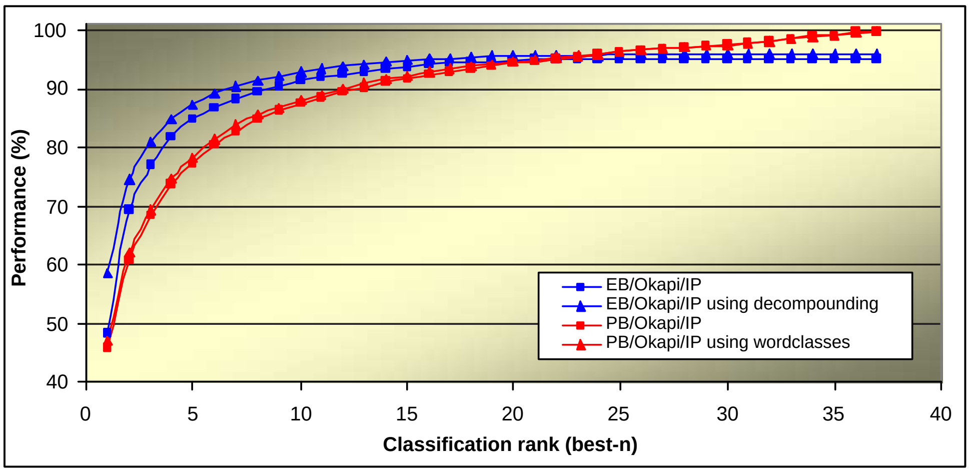

Figure 8.5: Influence of the application of language technology on the classification accuracy of the profile and example based classification methods

the EB/Okapi/IP classification method, and an increase of 2.58 percentage points with respect to the EB/Okapi/Cosine classification approach. This results in a total best-5 weighted classification accuracy of 87.4%. For the profile based classification approach, the use of language technology does not positively influence the classification accuracy for the best classification approach. For the other profile based classification approaches, the use of language technology slightly boosts the classification accuracy, but the total accuracy does not exceed the baseline accuracy of the PB/Okapi/IP combination, for which the parameter settings are optimal. Moreover, in case of the profile based classifier, the use of language technology can be compensated tuning the feature selection by setting the parameter ρ to a value for which the classification accuracy for this specific method reaches its maximum.

Chapter 9

Conclusions

In our first research hypothesis we stated that information retrieval (IR) based classification of e-mail messages for automatic answer suggestion, should outperform the manually defined keyword approach as developed by Em@ilco. In this thesis we presented two IR based classification approaches that certainly substantiate this hypothesis. In the classification routine as developed by Em@ilco, the correct answer was suggested in the first 5 suggestions in 40% of the cases. Our best performing IR based classification method, was able to more than double this performance: in 84.5% of the e-mails, the correct answer was suggested in the first 5 suggestions. So, we consider our first hypothesis as proven. It is worth noting that for almost 60% of all e-mails, the first answer was the correct answer, so automatic e-mail answering should be possible once we have a reliable confidence measure. This certainly encourages further research in this area.

The second research hypothesis stated that the use of language technology as text normalization, improves the accuracy of the IR based classification. In this thesis we have shown that this is partly true. Our example based classification approach (the K-Nearest-Neighbour classifier) benefits slightly from the use of language technology. The classification accuracy for this method increases with approximately 3 percentage points to a maximum of 87.36%, if we apply just one language technology (decompounding, stemming, language identification or POS-tagging in which we only use nouns, verbs and adjectives for classification, for best-5 weighted performance). If we apply combinations of two language technologies in the classification process, the increase in accuracy even approximates 4 percentage points, resulting in a maximum classification accuracy of 88.34%. However, for the profile based classification approach (the Rocchio classifier), the use of language technology does not significantly improve the classification accuracy.

The classification approaches presented in this thesis perform best if the documents (i.e. e-mails) contain as much relevant and as little irrelevant information as possible. With this we mean that an e-mail should not contain more information than the actual question the user wishes a response to, for optimal classification accuracy. The nature of the language technology we used in this thesis is twofold. We try to eliminate as much irrelevant terms as possible (removing stopwords or using only nouns, verbs and adjectives in the classification process). Moreover, we try to generalise the e-mails by relating different morphological variants of words and correcting spelling errors. In both matters we try to transform the e-mail into a representation as relevant as possible for the classification process (i.e. only the words that matter should be kept). Suppose we only receive e-mails in which the question has been reduced to a single sentence without customer information, e-mail signatures, reply lines, greetings, etc.. For instance, the e-mail contains no more than the sentence:

> *"How may I apply for more lottery tickets?"*

It should not be difficult to distinguish this question from the sentence:

> *"How may I cancel my lottery tickets?"*

However, if we pose the last question differently and provide a more realistic example with more (unfortunately irrelevant) information like the sentence below, discrimination will be more difficult:

> *"In august, I decided to apply for a lottery ticket, but now I realise that applying for this ticket was not a good idea! Therefore I wish to cancel it."*

If we want the system to classify the last e-mail in one of the categories representing the standard questions *"cancelling tickets"* and *"applying for tickets"*, it would probably be classified in the last, since the words *apply* and *applying* occur more often than *cancel*. The essence of this e-mail is that the user wants to cancel a ticket. However, even if we apply the language technology presented in this thesis, the system would not be able to classify this e-mail correctly.

If we wish to improve text classification systems using language technology, we have to strive for detecting and extracting the essence of a document: a very difficult problem, which may not adequately be solved by current state-of-the-art language technologies.

Overall we can state that the use of information retrieval based classification methods is certainly suitable for best-N answer suggesting in a contact centre. A good incorporation of the proposed technologies will certainly provide a reduction in time spent on answering e-mails, and thus on the investments that have to be made in implementing the technology.

Chapter 10

Suggestions for future work

No Master of Science thesis is complete without a bunch of useful ”future work” suggestions at the end. Moreover, we will point out some necessary requirements for the classification approach to perform optimally in a contact centre.

10.1 Research suggestions

In this research we demonstrated that in general, IR based classification methods for automatic e-mail answer suggestions perform good, and that classification accuracy is higher than that of support vector machines (Busemann et al., 2000) and Naive Bayes (Gaustad and Bouma, 2002). However, since the point of departure differs (i.e. the e-mail corpus), we can not decide which approach performs best for the presented cases. An evaluation study, like Yang and Liu (1999) conducted for text categorization on the Reuters-21578 corpus, should be done to compare such statistical based classification approaches for the e-mail domain.

10.1.1 Hierarchical classification

In this research we focused on using text classification techniques for answer suggestion on incoming e-mail in a contact centre. Although some contact centres are mono-subject (i.e. they work on one item for just one customer as is the case with the ”De Nationale Postcodeloterij”) some of them handle a wider variety of subjects (e.g. questions about telephony, television, and internet for a Telecom provider). Suppose we receive the following request:

> *”dear company, I want to complain about the internet service interruption. I told the lady on the telephone about it, but still I get no connection.”*

This message contains the important terms *internet*, *interruption* and *telephone*, which may be suggested an answer to a telephony service interruption question instead of the original internet question. Of course, different e-mail addresses can be used, but companies tend to use as few addresses as possible (more addresses means more communication and more communication means more money). To solve this problem, it is important to classify incoming mail first as belonging to one of the three broad classes (telephony, television, or internet).

In the second stage, the e-mail can be categorized within one of the clusters of the selected category. The advantage of such an approach might be that classification accuracy increases (since the classification problem has become easier due to the smaller category set). For each cluster an index (document collection) is created based on the messages that belong to that cluster. Such clusters do not necessarily have to be determined manually, but may also be determined by document clustering algorithms which cluster a set of documents in a set of categories.

10.1.2 Information extraction and deletion

The main theme of this thesis is that an e-mail may contain valuable information for classification and information retrieval systems, but that it can be difficult to determine which information is relevant. The important information, ”hidden” in many accompanying phrases, contains the actual questions, complaints, or remarks the customer wishes a response to. Such key-utterances are very difficult to detect, but we may decrease the complexity of the problem by removing utterances that are not relevant (personal information, greetings, welcome message, a signature below the e-mail message or reply-information). In general one can state that removing non-relevant information increases the relevance density of the remaining part of the message and so, increases the possibility of a correct classification.

Moreover, besides information extraction and deletion, we may also add information to the e-mail representation in our system by looking at the (structured) meta-information of an e-mail. E-mails contain information about the date, time sender, receiver and subject. Especially the subject might be an interesting addition for the classification because it usually specifies the content of the mail (although lots of people just reply on the last e-mail received).

10.1.3 Automatic answering

In the last decades, text classification mainly focussed on large text collections (e.g. news paper article collections) that can be categorized in relatively general topics (sports, economics, culture, foreign politics, etc). The research of Busemann et al. (2000), Scheffer (2004), and Gaustad and Bouma (2002) has demonstrated that text classification techniques perform well in suggesting relevant categories for a document collection containing small, unstructured documents in a relatively large (and detailed) set of categories. However, correct direct classification for such e-mail messages (best-1 performance) is not feasible due to the fact that differences between documents and the differences between categories are very small.

An interesting research topic might be automatically answering e-mail in contact centres. Based on the classification techniques presented in this chapter, we can focus on the 60% of the e-mails that can be automatically answered. Obviously, this 60% is not high enough to auto-answer all incoming e-mails automatically since 40% of the customers would receive an incorrect answer: additional confidence measurements are necessary!

Term weighting schemes in this thesis may provide us with useful information about the relative relevance of documents. Two categories with almost equal relevance scores are both good candidates for answer suggestion, but a category that has a relevance score that exceeds all of the relevance scores of the other categories, is more likely to be the correct category and thus can be used to answer the question. If the category suggestion is trustworthy enough, we might send it to the customer (with a note that this is an automatically generated answer). Suppose that 50% of the e-mails receive such ”one preferred category” and that 80% of the class selection of these e-mails is right. Then, we have the following situation:

80% x 50%	=	40% correctly responded
20% x 50%	=	10% incorrectly responded
50%	=	50% handled by human agents

If so, it should be interesting to see if the public will accept this: a nearly real time response at a cost of 10% erroneous answers.

10.1.4 Automatic parameter estimation

The presented classification approaches rely on correct parameter settings in order to achieve an optimal classification accuracy. In chapter 8 we stated that for each classification problem the parameters should be experimentally determined. For this research, the optimal value for K in K-Nearest-Neighbour classification turned out to be 50, and the optimal value for ρ in the Rocchio classification algorithm was 8. Moschitti (2003) experimented with an automatic parameter estimation algorithm. Based on a small reference set of the document collection, the parameter ρ for which the classification accuracy of a certain classification problem reaches its maximum, is automatically determined. For each classification problem, the value for ρ might be different. Investigating the use of such automatic parameter estimation algorithms based on the work of Moschitti for estimation of ρ and K may increase the usability of the classification approaches presented in this thesis.

10.2 Implementation suggestions

Text classification problems have in common that a given set of documents should be categorized within a given set of categories. The documents, categories and numbers naturally differ in different classification problems. For this reason we may not assume that the best classification and language technology combination presented here for the ”Nationale Postcode Loterij” are optimal too for a different classification problem. Even if the documents are comparable (both are e-mail sent to a service centre) and the number of categories is more or less equal, the optimal strategy may differ from the best in this report. Before implementing these classification algorithms in a contact centre, some experiments have to be conducted to determine the best classification approach for the specific set of documents and categories. Parameters like K in K-Nearest-Neighbour classification and α and β in Rocchio classification might have to be adjusted. Also, the application of language technology might not result in the same classification accuracy for different classification problems. The experiments to determine the optimal settings and classification approaches can be conducted using the prototype that we have developed for this research.

The effectiveness of these methods is entirely based on the nature of the classification problem and the training data. If the training data contains lots of classification errors, the approaches discussed in this thesis will inevitably copy this incorrect classification behaviour, resulting in a low classification accuracy and thus low usability in a contact centre. Also, the presence of tags for structuring information (e.g. to forward a web form using e-mail) can negatively manipulate the classification accuracy and should therefore be removed. Finally, text classification works best for a classification problem in which the categories are distinct and documents may be categorized in exactly one category. This implies that no categories should contain other categories. For example, if the contact centre uses categories named *"Email address - general"* and *"How may I change my e-mail address"*, these categories may overlap. This overlap introduces a problem to a classification algorithm since both categories contain mails with questions about the e-mail address, and contact centre agents will also categorize mails from the second category in the first one. The classification algorithm now finds two suitable categories for the same question.

In conclusion, before text classification approaches may prove their use in a contact centre, the classification organization first has to be optimalised. This implies the following:

- No double question/answer pairs (categories) may exist
- No overlapping categories may exist
- Equal questions requiring different answers (based on context information like a date, time or event) should be collected in the same category (since the questions are the same)

However, we expect that the combination of better classification approaches, more advanced language technologies and more structured e-mails will enable automatic answering of e-mails in the future.

Bibliography

I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras, and C.D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *ArXiv Computer Science e-prints*, 2000.

R.H. Baayen, R. Piepenbrock, and H. Van Rijn. The CELEX lexical database. linguistic data consortium, university of pennsylvania, 1993.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New York, 1999.

S. Bickel and T. Scheffer. Learning from message pairs for automatic email answering. In *ECML*, pages 87–98, 2004.

G. Boone. Concept features in re:agent, an intelligent email agent. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 141–148. ACM Press, New York, 1998.

E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21 (4):543–565, 1995.

J.D. Brutlag and C. Meek. Challenges of the email domain for text classification. In *Proceedings of ICML-2000, International Conference on Machine Learning*, pages 103–110, 2000.

C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 292–300, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

S. Busemann, S. Schmeier, and R.G. Arens. Message classification in the call center. In *Proceedings of the sixth conference on Applied natural language processing*, pages 158–165, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

W.B. Cavnar and J.M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US, 1994.

A. Chen. Cross-language retrieval experiments at clef 2002. In *Proceedings of CLEF-2002*, pages 28–48, 2002.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3): 273–297, 1995.

B.W. Croft. Knowledge-based and statistical approaches to text retrieval. *IEEE Expert*, 8(2):8–12, 1993.

H. Drucker, D. Wu, and V. Vapnik. Support vector machines for Spam categorization. *IEEE-NN*, 10(5):1048–1054, 1999.

D. Etzold. Improving spam filtering by combining naive bayes with simple k-nearest neighbor searches. *ArXiv Computer Science e-prints*, 2003.

T. Gaustad and G. Bouma. Accurate stemming of dutch for text classification. *Language and Computers*, 45(1):104–107, 2002.

J.M. Gomez, J.C. Cortizo, E. Puertas, and M. Ruiz. Concept indexing for automated text categorization. In *Proceedings of the 9th International Conference on Applications of Natural Languages to Information Systems, NLDB*, 2004.

D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(2):7–15, 1991.

D. Harman. Overview of the first trec conference. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 36–47, New York, NY, USA, 1993. ACM Press.

D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, 2000.

D. Hiemstra and F. de Jong. Statistical language models and information retrieval: natural language processing really meets retrieval. *GLOT International*, 5:288–294, 2001.

D. Hull. Stemming algorithms - a case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1), 1996.

T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. 10th European Conference on Machine Learning ECML-98*, pages 137–142, 1998.

T Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02)*, pages 132–142, New York, 1999. ACM Press.

D. Jurafsky and J.H. Martin. *Speech and Language Processing*. Prentice Hall, New Jersey, 2000.

M. D. Kernighan, K. W. Church, and W. A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the COLING-90*, volume 2, pages 205–211, Helsinki, Finland, 1990.

B Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *Proceedings of ECML'04, 15th European Conference on Machine Learning*, pages 217–226, 2004.

W. Kraaij and R. Pohlmann. Porter's stemming algorithm for dutch. In L.G.M. Noordman and W.A.M. de Vroomen, editors, *Informatiewetenschap 1994: Wetenschappelijke bijdragen aan de derde STINFON Conferentie*, pages 167–180, 1994.

W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 40–48, New York, USA, 1996. ACM Press.

J.B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

C. Monz and M. De Rijke. Shallow morphological analysis in monolingual information retrieval for dutch, german and italian. In *Proceedings CLEF 2001*, pages 262–277. Springer Verlag, 2001.

S. Mooers. Information retrieval viewed as temporal signaling. In *Proceedings of the International Congress of Mathematicians*, volume 1, pages 572–573, 1950.

A. Moschitti. A study on optimal parameter tuning for Rocchio text classifier. In Fabrizio Sebastiani, editor, *Proceedings of ECIR-03, 25th European Conference on Information Retrieval*, pages 420–435, Pisa, IT, 2003. Springer Verlag.

N. Oostdijk. The spoken dutch corpus. overview and first evaluation. In *Proceedings LREC 2002*, volume 2, pages 887–893, 2000.

M. Popovic and P. Willet. The effectiveness of stemming for natural-language access to slovene textual. *Journal of the American Society for Information Science*, 43(5):384–390, 1992.

M.F. Porter. An algoritm for suffix striping. In K. Sparck Jones and P. Willet, editors, *Readings in Information Retrieval*, 1980.

S.E. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33(4), 1977.

S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

S.E. Robertson and K. Sparck Jones. Simple, proven approaches to text retrieval. Technical report, City University London and University of Cambridge, 1997.

P. Rosso, E. Ferretti, D. Jiménez, and V.Vidal. Text categorization and information retrieval using wordnet senses. In *Proceedings of the second Global WordNet Conference*, pages 299–304, 2004.

G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, S.D. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6(1):49–73, 2003.

G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *Commications of the ACM*, 26:1022–1036, 1983.

G. Salton and M.J. McGill. *An introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

T. Scheffer. Email answering assistance by semi-supervised text classification. *Intelligent Data Analysis*, 8(5), 2004.

F. Sebastiani. A tutorial on automated text categorisation. In Analia Amandi and Ricardo Zunino, editors, *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, pages 7–35, Buenos Aires, AR, 1999.

K. Sparck Jones, S. Walker, and S.E. Robertson. A probabilistic model of information retrieval: Development and status, 1998.

C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

J. Yang and S. Park. Email categorization using fast machine learning algorithms. In *Discovery Science*, pages 316–323. Springer-Verlag, 2002.

Y. Yang. Expert network: Effective and efficent learning from human decisions in text categorisation and retrieval. In Bruce W. Croft and C. J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22. Springer-Verlag, 1994.

Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, USA, 1999. ACM Press.

Appendix A

Classification accuracy for NLP experiments

In this appendix we will present the classification accuracy for the experiments using language technology as a pre-process for the information retrieval based classification methods presented in this thesis. In chapter 8 we already presented the differences between the baseline experiments and the experiments using language technology, but in this appendix we present the total classification accuracy for the combinations of classification methods and language technologies. In each table, the classification methods are denoted in the rows and the language technologies in the columns. We use the following abbreviations:

Abbreviation	**Language technology**
sw	Stopword removal
dc	Decompounding
ng	N-gram check
st	Stemming
sp	Spelling correction
wc	Word classes (Part of speech tagging)

A.1 First experiment series

In this section we will present the classification accuracy of the first experiments series. In this experiment series we investigated the influence of each individual language technology on the classification accuracy. In table A.1 we present the classification accuracy of using language technology for the example based classification methods. In table A.2 we present the classification accuracy of using language technology for the profile based classification methods.

Experiment		**Weighted average performance (%)**						
n	**Approach**	**base**	**sw**	**dc**	**ng**	**st**	**sp**	**wc**
1	Okapi/IP	52.07	55.65	58.55	57.95	57.88	40.88	58.31
	Okapi/Cos	48.20	40.43	47.94	50.28	56.93	58.35	58.25
	TFIDF/IP	25.80	40.77	26.68	25.83	25.47	25.78	39.86
	TFIDF/Cos	47.25	40.69	46.93	36.12	53.87	54.69	57.82
3	Okapi/IP	74.63	77.73	81.05	80.71	80.59	61.88	80.64
	Okapi/Cos	77.15	73.46	77.49	78.10	78.05	79.24	78.61
	TFIDF/IP	43.11	63.23	43.96	43.30	42.16	43.25	60.66
	TFIDF/Cos	76.35	73.76	76.54	65.64	76.51	77.31	78.32
5	Okapi/IP	82.15	84.77	87.40	87.36	86.88	71.51	86.83
	Okapi/Cos	84.82	81.67	84.87	85.15	85.36	86.08	85.27
	TFIDF/IP	56.75	74.53	57.11	56.91	55.23	56.85	72.76
	TFIDF/Cos	83.89	81.87	83.96	76.31	84.18	84.63	85.30
10	Okapi/IP	89.47	91.47	92.96	92.91	92.85	83.44	92.62
	Okapi/Cos	91.42	89.37	91.59	91.90	91.69	92.05	91.67
	TFIDF/IP	71.78	85.17	72.21	72.01	71.63	72.33	84.03
	TFIDF/Cos	91.02	89.42	91.30	85.38	91.18	91.48	91.66

Table A.1: Classification accuracy of using language technology for the **Example based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

Experiment		**Weighted average performance (%)**						
n	**Approach**	**base**	**sw**	**dc**	**ng**	**st**	**sp**	**wc**
1	Okapi/IP	45.87	44.10	41.76	45.61	46.23	45.79	47.03
	Okapi/Cos	27.49	27.95	25.73	27.78	26.07	27.31	28.53
	TFIDF/IP	43.07	42.45	36.87	42.56	43.17	43.01	45.40
	TFIDF/Cos	27.20	27.22	25.14	27.49	25.49	27.03	27.78
3	Okapi/IP	68.45	67.67	64.73	68.18	68.21	68.35	69.51
	Okapi/Cos	52.39	54.53	48.77	52.54	51.66	52.18	54.52
	TFIDF/IP	67.11	67.12	62.65	66.90	67.17	67.21	68.84
	TFIDF/Cos	50.62	52.76	46.71	50.98	49.76	50.61	52.80
5	Okapi/IP	77.40	77.38	74.71	77.26	76.84	77.24	78.31
	Okapi/Cos	64.09	66.39	60.92	64.19	68.29	63.84	66.07
	TFIDF/IP	76.50	76.78	73.40	76.55	76.20	76.45	77.81
	TFIDF/Cos	63.08	65.73	59.29	63.06	62.91	62.95	64.93
10	Okapi/IP	87.51	87.91	85.82	87.46	87.05	87.33	88.07
	Okapi/Cos	80.53	82.43	77.77	80.32	80.24	80.17	81.69
	TFIDF/IP	87.21	87.69	85.57	87.36	86.81	87.08	87.89
	TFIDF/Cos	80.02	82.11	77.40	79.98	79.94	79.73	81.48

Table A.2: Classification accuracy of using language technology for the **Profile based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

Experiment		**Weighted average performance (%)**						
n	**Approach**	**wc/st**	**sw/dc**	**sw/ng**	**dc/st**	**sw/st**	**dc/wc**	**ng/st**
1	Okapi/IP	46.57	56.06	56.46	57.26	59.23	58.00	58.18
	Okapi/Cos	57.77	54.83	47.96	55.77	58.29	57.81	51.92
	TFIDF/IP	30.79	41.47	41.32	25.75	42.85	40.22	25.48
	TFIDF/Cos	57.34	54.12	47.51	53.06	57.83	57.60	50.72
3	Okapi/IP	64.27	79.35	79.07	80.87	81.47	80.74	80.90
	Okapi/Cos	78.04	76.59	75.36	77.82	79.37	78.71	77.81
	TFIDF/IP	46.42	62.65	64.25	43.15	67.16	59.68	42.12
	TFIDF/Cos	77.99	76.50	75.25	76.30	79.37	78.35	76.53
5	Okapi/IP	69.36	85.93	85.88	87.06	88.34	86.93	86.96
	Okapi/Cos	85.05	84.05	82.83	85.13	86.50	85.33	84.89
	TFIDF/IP	55.70	73.99	75.27	56.92	78.33	71.94	55.10
	TFIDF/Cos	84.92	83.97	82.71	84.12	86.68	85.36	84.11
10	Okapi/IP	74.19	92.07	92.18	92.92	93.33	92.72	92.88
	Okapi/Cos	91.59	90.71	89.79	91.64	92.52	91.54	91.39
	TFIDF/IP	65.43	85.16	85.74	72.49	87.90	83.38	71.62
	TFIDF/Cos	91.74	90.73	89.94	91.21	92.65	91.48	91.10

Table A.3: Classification accuracy of using binary combinations of language technology for the **Example based** methods. The first column lists the value for n to measure the best-n performance in weighted averages

A.2 Second experiment series

In this section we will present the classification accuracy of the second experiments series. In this experiment series we investigated the influence of binary combinations of language technology on the classification accuracy. In table A.3 we present the classification accuracy of using language technology for the example based classification methods. In table A.4 we present the classification accuracy of using language technology for the profile based classification methods.

A.3 Third experiments series

In this section we will present the results for our third experiments series. In table A.5 we present the total classification accuracy for applying combinations of three language technologies on the example and profile based classification methods.

Experiment		**Weighted average performance (%)**						
n	**Approach**	**wc/st**	**sw/dc**	**sw/ng**	**dc/st**	**sw/st**	**dc/wc**	**ng/st**
1	Okapi/IP	43.59	39.91	43.32	42.46	42.51	42.51	45.83
	Okapi/Cos	26.93	25.98	28.05	23.85	26.23	26.45	26.52
	TFIDF/IP	41.61	36.05	41.88	38.48	41.68	38.40	42.82
	TFIDF/Cos	26.60	24.67	27.59	23.82	26.00	25.63	26.03
3	Okapi/IP	66.60	63.44	67.24	65.04	67.79	66.12	68.01
	Okapi/Cos	54.51	50.15	54.24	47.59	49.67	50.81	52.36
	TFIDF/IP	65.94	62.09	66.46	63.14	67.65	64.42	67.04
	TFIDF/Cos	52.59	47.70	52.66	45.25	48.09	48.73	50.62
5	Okapi/IP	75.85	73.53	76.92	74.06	77.35	75.82	77.15
	Okapi/Cos	66.83	62.23	66.14	60.44	61.86	63.13	64.53
	TFIDF/IP	75.61	73.39	76.45	73.28	77.35	75.09	76.60
	TFIDF/Cos	66.01	61.55	65.43	58.97	61.29	61.50	63.79
10	Okapi/IP	86.39	85.58	87.87	85.08	87.61	86.29	87.17
	Okapi/Cos	82.05	78.74	82.24	77.68	78.56	79.54	80.79
	TFIDF/IP	86.39	85.76	87.88	85.09	87.73	86.15	87.25
	TFIDF/Cos	82.01	79.26	82.02	77.39	78.92	79.17	80.84

Table A.4: Classification accuracy of using binary combinations of language technology for the **Profile based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

		Weighted average performance (%)			
Experiment		**Example based**		**Profile based**	
n	**Approach**	**sw/dc/st**	**sw/ng/st**	**sw/dc/st**	**sw/ng/st**
1	Okapi/IP	56.12	41.41	38.68	41.68
	Okapi/Cos	54.36	47.52	24.21	25.53
	TFIDF/IP	41.25	40.50	36.99	40.71
	TFIDF/Cos	54.87	47.87	23.617	26.04
3	Okapi/IP	78.89	62.06	63.39	67.37
	Okapi/Cos	76.18	75.01	48.31	53.72
	TFIDF/IP	61.92	61.04	63.13	67.37
	TFIDF/Cos	76.36	75.04	46.44	50.62
5	Okapi/IP	85.84	67.95	73.54	78.17
	Okapi/Cos	83.79	82.80	61.728	63.95
	TFIDF/IP	73.42	67.24	73.91	78.25
	TFIDF/Cos	83.96	82.66	61.19	63.76
10	Okapi/IP	91.77	73.11	84.99	87.68
	Okapi/Cos	90.77	89.79	78.28	82.17
	TFIDF/IP	84.61	72.90	85.71	87.84
	TFIDF/Cos	90.98	90.04	78.99	82.37

Table A.5: Classification accuracy of combinations of three language technologies for the **Profile based** methods. The first column lists the value for *n* to measure the best-*n* performance in weighted averages

Appendix B

Wordclasses of the POS-tagger

The lingware tool-kit developed by Carp Technologies uses the following Word Classes in the Part of Speech tagger:

Tag	**WordClass**	**Dutch translation**
ADJ	Adjective	Bijvoeglijk naamwoord
ADJC	Comparative adjective	Vergelijkend bijvoeglijk naamwoord
ADJS	Superlative adjective	Overtreffend bijvoeglijk naamwoord
ADV	Adverb	Bijwoord
CONJ	Conjunction	Voegwoord
DET	Determiner	Determinator
END	End of sentence	Zinseinde
INTR	Interjection	Tussenwerpsel
NS	Noun singular	Enkelvoudig zelfstandig naamwoord
NP	Noun plural	Meervoudig zelfstandig naamwoord
NUM	Numeral	Nummer
PMK	Punctuation mark	Interpunctie
PREP	Preposition	Voorzetsel
PRON	Pronoun	Voornaamwoord
PROP	Proper noun	Eigennaam
PRONDEMO	Demonstrative pronoun	Aanwijzend voornaamwoord
PRONPERS	Personal pronoun	Persoonlijk voornaamwoord
PRONRELA	Relative pronoun	Betrekkelijk voornaamwoord
PRONQUES	Interrogative pronoun	Vragend voornaamwoord

Tag	WordClass	Dutch translation
PRONPOSS	Possessive pronoun	Bezittelijk voornaamwoord
PRONWKND	Reflexive pronoun	Wederkerend voornaamwoord
Q	Quantifier	Kwantor
UNK	Unknown	Onbekend
VAI	Verb auxiliary infinitive	Hulpwerkwoord, onbepaald
VAPAS	Verb auxiliary past	Hulpwerkwoord, verleden tijd
VAPASPAR	Verb auxiliary past participle	Hulwerkwoord, voltooid deelwoord
VAPRE	Verb auxiliary present	Hulpwerkwoord, tegenwoordige tijd
VAPREPAR	Verb auxiliary present participle	Hulpwerkwoord, tegenwoordig deelwoord
VI	Verb infinitive	Onverbogen werkwoord
VPAS	Verb past	Werkwoord, verleden tijd
VPASPAR	Verb past participle	Voltooid deelwoord
VPRE	Verb present	Werkwoord, tegen-woordige tijd
VPREPAR	Verb present participle	Tegenwoordig deelwoord

Table B.1: Wordclasses used in the Part-of-Speech tagger developed by Carp Technologies

Appendix C

Lingware toolkit output

This appendix contains a listing of the output generated by the Lingware tool-kit output developed by Carp Technologies. We present the results for the sample Dutch sentence: *Ik koopp een fiets* (English: *I buy a bicycle* with a spelling error in the verb *buy*). The output is presented in an XML format, in the container `<reply> </reply>`. The results per token are tab-delimited.

```
<reply>
        TokenIndex:  0
           Original token: ik
           Used token:           ik
           CLASS:                SCRIPT
           Word frequency:       481518
           Word classes:         PRONPERS (1.0)
           Stems:
           Word parts:           ik (1.0)
           Preferred spellings:  ik (1.0)
        TokenIndex:  1
           Original token:       koopp
           Used token:           koop
           CLASS:                SCRIPT
           Word frequency:       0
           Word classes:         VPRE (1.0)
           Stems:                kopen (1.0)
           Word parts            kopen (1.0)
           Preferred spellings:  koopt (kopen) (0.8)
```

```
          TokenIndex:  2
             Original token:       een
             Used token:           een
             CLASS:                SCRIPT
             Word frequency:       1079982
             Word classes:         DET (1.0)
             Stems:
             Word parts:           een (1.0)
             Preferred spellings:  eer (0.5), pen (0.5),
                                   eed (0.5), wen (0.5),
                                   hen (0.5), en (0.5),
                                   eet (0.5), ene (0.5),
                                   gen (0.5), ren (0.5),
                                   jen (0.5), ken (0.5),
                                   den (0.5), ben (0.5),
                                   ven (0.5), eek (0.5),
                                   sen (0.5), oen (0.5),
                                   ten (0.5), een (1.0),
                                   men (0.5)
          TokenIndex:  3
             Original token:       fiets
             Used token:           fiets
             CLASS:                SCRIPT
             Word frequency:       1768
             Word classes:         NS (1.0)
             Stems:                fiets (1.0)
             Word parts:           fiets (1.0)
             Preferred spellings:  fiets (1.0)
</reply>
```

The explanation of the Lingware tool-kit output is presented in table C.1. For every feature that may contains different values (because of ambiguity a word may be matched to several stems and word classes), the probability is given for each option and ranges from 0 to 1.

Feature	**Explanation**
Original token	The token that is read directly from the input (no change has been made)
Used token	This is the token that is used for further processing, if the word spelling was incorrect, the spelling is corrected
Word frequency	The number of occurrences of the original token in the corpus
Word classes	The possible word classes that fit the token. The probability for each word class is presented between the braces (and ranges from 0 to 1)
Stems	The stem of the used token (if known). Nouns are stemmed to their singular form and verbs are stemmed to their non-conjugated form.
Word parts	Lists the word parts of a compound
Preferred spellings	A list of possible spellings that may be suggested by a spelling checker if the used token is misspelled. This is not a spelling correction routine, but a spelling suggestion routine

Table C.1: Explanation of the Lingware tool-kit output