Piet van Delft

June 2005

Master Thesis Business Information Technology

University of Twente

Graduation committee:

prof. dr. F.M.G. de Jong

dr. ir. J. Kuper (1st supervisor)

dr. R.E. Leenes

prof. dr. R. Marcinkevičienė

Cover art by Frans van Delft

based on the painting ‘De toren van Babel’ by Pieter Bruegel de Oude, ca 1563

Preface

This report and the software that can be found on the cd-rom accompanying the printed version, form the master thesis from my studies Business Information Technology at the University of Twente in the Netherlands. I have found this master project to be very interesting and the whole experience of working on it a pleasure, the latter being not in the last place reflected by the timespan of a whole year that it took me to finish it. But I dare anyone to fly to Lithuania, see its beauty, meet all the interesting people I've met and still get a good day's work done. The same goes for that half year in Amsterdam! And actually my entire period in Enschede... okay maybe it's just me. Nevertheless it was never the thesis to blame, because working on it was sometimes hard but never tedious and I have enjoyed learning about linguistics in general and the Lithuanian language especially, in combination with Law, the European Union, and information retrieval as well as picking up some of that C++ for the first time.

I would like to thank all of my supervisors for their bravery of having their names on the front of this report for everyone to see. Jan Kuper, for letting me do whatever I wanted to, but still coming through with solid advice when I needed it, Rūta Marcinkevičienė, for putting up with the wilfulness of this Dutch student and providing me with all the facilities at the CCL in Kaunas, and Ronald Leenes, for his time and advice on voluntary basis as he is no longer connected to the University of Twente as a professor.

Furthermore I would like to thank Ralf Steinberger and Bruno Pouliquen from the JRC for providing the corpora and insight in their research. Of course, I could not forget to thank Juratė and Ugnė enough for their unsurpassed patience in trying to teach me some Lithuanian and I am very sorry to say I still speak better Smurf. Thanks also to my Mum and Dad and my brother for supporting me wherever I am and whatever I'm doing and all my friends who tried to make me graduate faster and all those that did their best to never make me graduate at all, each of them with their own good intentions. You've all been brilliant!

Piet van Delft

Amsterdam, the Netherlands, June 20th 2005

Summary

The European Union is becoming larger and more important every day and to keep things running is hard enough as it is, without the obstacle of having to deal with the twenty different official EU languages. The EU provides a lot of work for translators, who not only translate reports and brochures, but also national and European legislation. Translating legislative documents in particular is a very complex task of the utmost importance. The language of law is a field of knowledge on its own for a translator and is different for each language, in fact even the European Union as a whole can be seen to slowly develop its own judicial language. For their task translators need more than a dictionary, they need to be able to draw from references in the form of similar documents in different languages.

To keep the enormous document archives of the national and international parliaments of the European Union organised and accessible, the Publications Office of the EU has developed the Eurovoc thesaurus. This thesaurus is an organised list of some 6500 terms which are to be used as keywords for describing the topic of a document and which are attached to the document as metadata. Because this list is translated into all official languages of the EU, it is possible to search for these Eurovoc keywords in one language and by doing so also retrieve documents in other languages.

Attaching these keywords to documents, called *indexing*, is sheer drudgery and therefore ways are sought to lighten the task with the use of information technology. The Joint Research Centre of the EU is working on such a technology for automatic indexing with Eurovoc keywords. Their system works by first training it on a large collection of documents which have already keywords assigned to them, this way the system learns which types of documents go with which keywords. Although this methodology works independent of the language of the text, research from the JRC has shown that language-specific pre-processing can mean an improvement to the results.

For the Lithuanian language, and for other languages with rich morphology, an improvement can be accomplished by pre-processing documents with a stemmer which strips the suffixes from nouns, adjectives, and pronouns. This stemmer need not be grammatically correct in its stemming per se.

More improvement can be established by segmenting the text into its predefined structural components, as there are: title, preamble, chapter headings, and appendices. This segmenting can be accomplished by searching for key phrases in the text that indicate the start of such a segment. When each of these segments is treated separately and is given a different weight in our calculations, the title, preamble and appendices all appear to contain more usefull information than the rest of the text and accounting for this in our calculations gives us a small but sure improvement. This technique could be extended to other languages as well.

Finally, there is a possibility to automatically, without the use of key phrases, segment the largest part of the text, and by a second iteration during the calculations establish the relative importance of each of the found segments. This way a text could be separated into more and less important segments and we could give more weight to the important segments in our calculations. However the viability or potential gain of this hypothesis is at this point untested.

Table of contents

Glossary

Thesaurus	A controlled list of terms to be used in categorising large collections of texts. The list of terms is often hierarchically ordered and can have additional mutual relations appointed to the terms, like synonymy, hyponymy and other associations.
Descriptor	A term that is assigned to a text and describes the subject of the text, in the case of eurovoc the descriptors are selected from the terms in the thesaurus. A text can have multiple descriptors.
Associate	A term that is coupled with a descriptor. If an associate is found in a text then it is an indication that its descriptor can be assigned to this text. Each descriptor has a long list of associates.
Weight	A number that can be assigned to a variable to signify its relative importance. If x has a weight of 0.2 and y has a weight of 0.1, then x is twice as important as y. In a mathematical algorithm this is often called a coefficient.
Indexing	The process of assigning descriptors to a text (automatically or manual) that are representative of its subject.
Corpus	A large collection of texts used in linguistic research.
Recall	The fraction of the relevant items which have been retrieved.
Precision	The fraction of the retrieved items which are relevant.
Inflection	a change in the form of a word (usually by adding a suffix) to indicate a change in its grammatical function.
Stop list	A list of the most commonly used words in a language or a subset of a language.
Stemming	reduction of morphological variants of a word to a common stem.
Morpheme	The smallest part of a word that still carries meaning. e.g. 'birds' has two morphemes: 'bird' and 's', the first has an obvious meaning and the second adds the plural, it means there is more than one bird.
Lexical gap	A concept in one language that cannot be translated to a single term in another language causes a lexical gap in the second language.
Phonology	The area of linguistics that deals with the systematics of sounds and their relation to grammar.
Register	Subset of a language used for a particular purpose or in a particular social setting.
Semantic overlap	When a word in one language shares one meaning with a word in another language, but does not share all of the possible meanings of the second word. Only in some cases, it can be a correct translation.

Granularity mismatch	An inconsistency in classification of terms between two languages.
Lexicon	A vocabulary; a collection of the words in a language.
Precedent	In law, a precedent is a conclusion that is drawn in a case, which has an impact on the way similar cases will be handled. If the precedent becomes mandatory in future cases, it is called a *binding precedent*.

1 Introduction

1.1 European languages and Eurovoc

The allegory of the European Union as a second tower of Babel has not passed by many a person that is interested in linguistics within an EU context. Both can be considered a united and ambitious attempt to build something great, but where the builders of the tower of Babel had to give up their efforts because of their divine punishment with different languages and their subsequent inability to communicate and work together, the people building the EU are trying to overcome the language barriers by all sorts of means. They try to speak each other's languages or agree on a common lingua franca — which in the early years was faithful to the term the French language, but in the course of time this has been gradually replaced by English — or they avoid the need to know the same language by the use of translators in text and speech. More lately, the EU has also shown interest in the possibilities that modern technology have to offer in crossing the language barriers.

One of the initiatives of the EU is the Eurovoc project. The core of this project is the Eurovoc thesaurus — more on thesauri can be found in chapter 2 — which, put simply, is a list of about 6500 terms — called **descriptors** — translated in all EU languages. The idea behind Eurovoc is that each of the documents that are used and produced by national and European governments is tagged with a number of these terms from the list in order to describe its contents. For instance, a document on the environmental consequences of tuna-fishing in Hawaii gets tagged with the following terms:

fisheries policy
deep-sea fishing
hawaii
coastal pollution
fishery research

The use of these terms is universal and mandatory, meaning that nobody would use *sea fishing*, *high-seas fishing*, or even *tuna fishing* instead of *deep-sea fishing*, nor would they use something like *Maui* or *the Hawaiian islands*, but always strictly *Hawaii*. Because of this agreement on terms to use for describing a document it becomes more easy to browse and search in the huge collections of documents that are available. Also, because these terms are translated into other languages it even becomes possible to search in documents in different languages at the same time, or even in a language the user is unfamiliar with. The term *fisheries policy* is in the Dutch version of the Eurovoc thesaurus *visserijbeleid* and in the Lithuanian version *žvejybos politika*. Now, when an English speaking person uses a search-engine that works with the Eurovoc thesaurus and queries for documents on *fisheries policy*, the search engine, which can link the English version of this term to all the versions in other languages, is capable of not only returning the English documents on the topic, but also those in other languages. Of course at that point, the English user still cannot read the entire document in Lithuanian or Dutch, but if the document looks like something usable at first glance or perhaps based on an English summary, the person could proceed to have it translated or otherwise request more information on the document.

1.2 Language Technology at the Joint Research Centre

The JRC is the Joint Research Centre of the European Commission, it defines itself as follows:

> *"The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union...". [14]*

Research is being conducted by the JRC in the field of computer-linguistics with the ultimate goal to *"give users cross-language access to information 'hidden' in large amounts of multilingual text".* One of their projects is 'cross-lingual indexing', which concerns automating the assignment of terms in the Eurovoc thesaurus to individual documents independent of their language.

Tagging documents with terms from the Eurovoc thesaurus can be a great help in archiving and retrieving them, but it is also a laborious task and in some cases the choices between different terms is not always a clear-cut one. This is why the technology worked on at the JRC of automating the process of assigning terms to documents could be helpful, either as a fully-fledged replacement for the manual task or as a tool to assist the human expert. The process of automatically assigning terms to documents as implemented by the JRC is discussed in chapter 6. Although the basic technology is language-independent, i.e. it can be applied to a document in any of the European languages without modifications, it is possible to improve the results by adjusting it for a specific language.

1.3 Linguistic research at the CCL

The CCL is the centre for Computational Linguistics of the Vytautas Magnus University in Kaunas, Lithuania. The goal of the centre is to provide and maintain a corpus and other linguistic resources for the Lithuanian language as well as software tools for corpus analysis. The CCL also does their own research into the Lithuanian language, for example into the Lithuanian translation of the Bible or extensively comparing the English and Lithuanian versions of the book '1984' by George Orwell, thereby adding to the scientific insights into this fascinating language. The JRC has asked the CCL for the use of their resources and knowledge in assistance of incorporating the Lithuanian language into their research.

1.4 Thesis research

The research of this thesis is aimed at exploring and where possible implementing the possibilities for improving the indexing technology by the JRC for the Lithuanian language. To this end I have researched the possibilities of the specific rich morphology of the Lithuanian language resulting in an inflectional stemmer, which has brought some good results in terms of improvement. Furthermore, I have researched if and how using the specific predefined formal structure of the legislative texts can be a source for improvement, resulting in experiments with identifying different segments of the text and using this extra data during the automated assignment of descriptors. This has also led to some improvements albeit less than those achieved with the stemmer. But before I will get to these experiments and their results, I will first dive into the main concepts of this thesis, namely the Eurovoc thesaurus, legislative documents, the research done at the JRC and the Lithuanian language, starting of with the Eurovoc thesaurus.

2 Thesauri

2.1 Standard thesauri

Constructing a thesaurus is more than accumulating the most often-used terms and organising them in groups, it is perhaps best illustrated as building an ontology of its context, in this case European legislation. To begin with, the thesaurus has to be complete, covering all possible concepts that appear in the texts, and secondly it has to define all the relevant semantic relations that these concepts can have amongst each other. These relations are not always clear and form the most difficult obstacles to overcome in constructing a thesaurus. Roughly they can be split up into four categories: **polysemy**, **homonymy**, **synonymy** and **hyponymy**.

Polysemy is when one word has multiple related meanings, for example the word 'play' can either be a game or a theatre show. *Homonymy* is when two unrelated words happen to have the same appearance, an example would be the words 'bank' as in the place one would take one's money and 'bank' as in the side of a river[1]. For both polysemy and homonymy the actual meaning of the word can only be concluded from the context of the word, i.e. we would at least have to see the entire sentence or more to choose between the different meanings. For this reason caution has to be taken in choosing and defining terms to prevent ambiguity in the thesaurus. In the case of Eurovoc so-called 'Scope Notes' are added to a term to explain more about its meaning or use when deemed necessary.

The third category — *synonymy* — is when two different words have the same meaning, as for instance 'complex' and 'complicated' or 'glad' and 'happy'. The trouble with synonyms is that if we would include all synonyms of every term the thesaurus will become bloated and it defeats the object of creating a taxonomy with a single term for a single concept. If, on the other hand, we do not include any of the synonyms we deny the fact that people use different words to say the same things and we run the risk of creating an unusable thesaurus. The solution is, as was also done for Eurovoc, to appoint one synonym as the 'official term' to be used in categorising documents, but still mention the other synonyms with it in the thesaurus.

The last category — *hyponymy* — is probably the most difficult one. Hyponymy defines a hierarchical relation between words, for instance apple is a hyponym of fruit and fruit in its turn is a hyponym of food, in other words: A is a hyponym of B if 'A is a type of B'. The thesaurus is ordered into a hierarchical system based on this type of relation, but in this process there are quite some difficult decisions to be made because in natural language hyponymy is everything but a straightforward system. To give an example, a cow is a type of animal, but it is also — for most people anyway — a type of food, so should we place it under food or under animals or both? Again, if we just place it under every category we can possibly think of — herbivore, mammal, female, et cetera — the thesaurus will become bloated and this is not what we want, because we want it to represent a certain **lexicon** — a certain vocabulary — but with as few complications of natural language as possible. In this case the solution is not as clear-cut as with the synonyms, so we will have to decide for each case individually what to do, which can quite often lead to arbitrary or even unsatisfactory choices. These difficulties do not only arise for single terms but also for entire classes of

[1] as opposed to the meanings of bank as a company on the one hand and a building on the other hand, which would be polysemy because the meanings are closely related.

terms which can be hard to place in the hierarchy or for instance are mutually overlapping, when this happens we also have to make a decision for each specific case.

We see that polysemy and homonymy introduce nothing but confusion, synonymy confronts us with some decisions to be made, but helps us reduce the number of terms in the thesaurus, and hyponymy is the basis for the hierarchical structure of the thesaurus, but comes with its own set of problems.

2.2 Multi-lingual thesauri

A multi-lingual thesaurus actually consists of multiple thesauri — one for each language — that are linked together, they share the same concepts and the same structure and can be considered each others translations. But there is more to its construction than just building one thesaurus —for instance an English one — and then getting a French and Spanish dictionary and starting to translate all the individual terms. Translating between languages in general is more than transposing the words with the help of a dictionary and a 'grammar for dummies' book, because of the idiom, i.e. the overtone of the cultural context in which a language is used. Although the terms in a thesaurus are secluded from a verbal context they still carry the cultural context from the person who put them there — that person could have also chosen to assign it to a different category or to use a different word at all to express a concept — and are again put into a context by the person who uses and interprets the thesaurus. Take for instance the word 'coffee shop', in most countries this would be a place where they sell coffee, in the Netherlands it is a place where soft drugs is being sold — and perhaps also coffee. There can also be less obvious differences that are hard to translate, for instance the difference between petty crime, 'normal' crime and capital crime is entirely dependent upon the law in a country, the words themselves are easily translatable, but their meanings vary between countries and languages.

Apart from these culture-based differences between a word and its literal translation, there are some cases in which translating a concept into another language in one single term is not possible at all, this is called a **lexical gap**. Often when this occurs the gap is filled with a loan word — i.e. we just start using the foreign word — this happened for words like 'glasnost' (Russian) or 'hangar' (French), but this is not always done and that leaves us sometimes with a translation problem, as happens with the English 'debriefing' or the well-known Dutch word 'gezellig'. A similar phenomenon is when two words in different languages don't match for the full 100 percent, like the French 'fille' and the English 'girl' — 'fille' can also mean 'daughter'. When this happens we speak of a **semantic overlap**. Luckily these cases are not that common.

There is one other possible mismatch between languages which relates to hyponymy and it occurs when there is a **granularity mismatch**. Take for instance the following example concerning the classification of monkeys in English and Dutch:

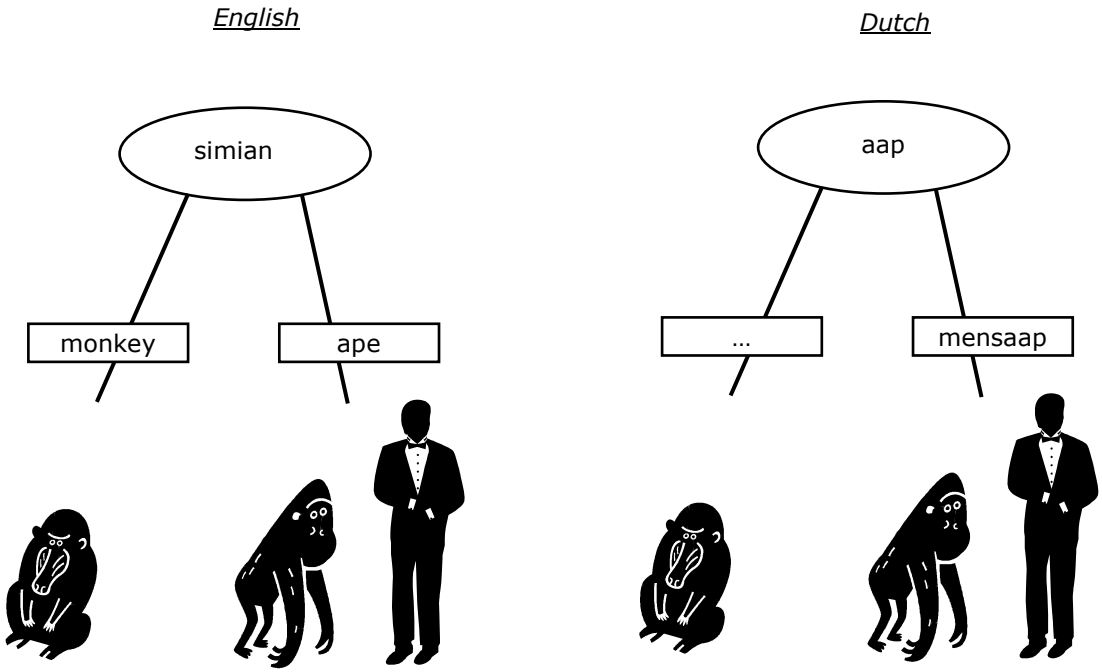


Figure 1: granularity mismatch between English and Dutch

The crux of the matter is the fact that Dutch does not have a distinctive word for the group of primates that *are* simians but *are not* apes. Take for instance the mandrill, the one with the long snout on the left in Figure 1. It's a monkey. In Dutch this would become 'aap', which is a perfectly valid translation that can be used in any common translation of one text to another. When we are building a multi-lingual thesaurus however, we do not only seek a correct one-on-one translation but also a correct classification as the thesaurus is hierarchically organised. As soon as we translate 'monkey' to 'aap' we actually lose the level of detail that the English term gave us. Saying in English that a mandrill is a monkey places it in a different group than humans, but saying in Dutch that a mandrill is an 'aap' places it in the same group that also humans belong to. This phenomenon is sort of a lexical gap, except that it does not pose a problem in everyday translations, but only comes to surface when we are dealing with classification like we are when building a thesaurus.

2.3 Figures of the Eurovoc thesaurus

Eurovoc is a multilingual thesaurus that resulted from a project launched in 1980 by the European Parliament and the Office for Official Publication of the European Communities to assist in indexing the large number of documents used and produced by the European parliaments. Eurovoc currently contains about 6500 descriptors which are organised in a hierarchical system with a maximal depth of eight levels. There are also lists of non-descriptors which are synonym to the descriptors but are not to be used, for instance in the example given on page 11 *Hawaii* is a descriptor and *Hawaiian Islands* is a non-descriptor. The amount of non-descriptors varies per language and ranges from 5333 (Finish) to 9257 (Italian). The Eurovoc thesaurus has been translated into 21 languages and is currently in use by the European Parliament and several National Parliaments and governmental organisations in the European region.

3 Europe

The European community was — in the form of its predecessor the European Coal and Steel Community — and still is today in essence an economic alliance. Although in its new treaty for the establishment of a constitution we can see the foundations being created for a shared foreign policy and even the first signs of a European military force, most agreements made so far and probably for some time to come deals with economic matters. The focus is still on Europe as a free marketplace and not Europe as a federation of states. A direct consequence of this is that the European legislation and also the cases that come before the European Court of Justice do not deal with all the same topics as the government and courts on a national level. For instance criminal law — except large-scale international crime — or national administrative law are two issues where national sovereignty is preserved. The general principle behind what should or should not be handled by the EU lies in the principle of subsidiarity, which says that what the lower entity in the hierarchy can do adequately should not be done by the greater entity unless it can do it better. In effect, this means that everything is left over to the national level unless doing it on a European level will mean that it is done more effectively. Ideally speaking of course, because the political reality seems more capricious than that, but even so the big issues in Brussels are the movement of people, goods, services, and capital, i.e. private law. Also the scientific discourse is centred on a European Ius Commune focussing on private law.

Any legislation coming from Brussels is published in the 20 official languages of the EU. Because citizens have the right to be able to communicate with the EU administration in their own language — see article I-10(2)(d) of the constitution treaty[1] — all official languages of the member states are considered to be official languages of the EU, treaties will be translated into all languages and all these translations have the same legal force. Also all the members of parliament have the right to speak in their own language. As a result of this policy there is a massive amount of translation services available to the Union. In 2003 the European committee employed 1.150 translators, together translating about 1,1 million pages and a further 300.000 pages were translated by freelance translators. The European Parliament employed 240 full-time interpreters and has another 1.000 interpreters on call for conferences [8]. With the recent addition of ten new member states these figures will be even bigger by now. In spite of the possibility for members of parliament to speak in their own language English is still the lingua franca of the EU and all treaties — which form in fact the European legislation — will mostly have been negotiated and for the first time written down in English. But the English spoken in Brussels is an international type of English interlarded with diplomatic jargon different from everyday British or American English. As any other professional group the diplomats have developed there own lingo in their everyday work.

The very same goes for the Court of Justice — sole supplier of European jurisprudence — and any other party that is somehow involved in the creation of EU law, like scientists or EU committees. Together all these parties form the first and highest authority responsible for developing a European system of law and because of this they are not bound to any existing lexicon and are to some extent free in shaping the new language of European law. Of course, the existing legislation in the EU will heavily influence the new legislation, but

[1] Those that question the validity of this constitution treaty due to recent events can rest assured that the right to communicate in one's own language with the EU administration has been around since the treaty of Rome, which was signed in 1958.

a new lexicon will take shape none-the-less because a) EU law introduces new issues unseen in national law and b) there is no single source from which regulations and their accompanying lexicon are adopted.

In a sense the development of EU legislation — consisting of all treaties and the jurisprudence of the Court of Justice — is not an effort of integrating existing legislation at the text level but at a political and administrative level resulting in a document — a treaty, a protocol, a case text, et cetera — in one single language. Only at that point the challenge starts of translating it into other languages or incorporating it into other law systems. Sometimes however it is deliberately chosen to look at a textual level to existing legislation as was done for instance with the text for some of the basic rights in the treaty for the constitution. In this case, the European Council had asked the state leaders of the European Committee to see to it..

> *"..that this Charter should contain the fundamental rights and freedoms as well as basic procedural rights guaranteed by the European Convention for the Protection of Human Rights and Fundamental Freedoms and derived from the constitutional traditions common to the Member States.." [5]*

In effect the Council asked the composers of the treaty to go and look at the way these basic rights were phrased in all the constitutions of the member states and to try to infer from this one common European phrasing.

These kind of cases however are an exception and on average we can say that the contents of documents is discussed upon and composed in one language and then translated to many other languages. In this sense European legislative documents are just as unique with respect to content and linguistics as legislative documents from any of its member states.

4 When laws meet

Most of us — especially those who live in a big city — know that in today's globalised world all sorts of languages can meet each other just around the corner of the street, in a local shop, a bar, or the park. While it is not as common as *that* for the laws of two nations to meet, there are still some circumstances under which laws meet — and have met for ages — and this is when international trade takes place. Because of this, the most research that is being done in legal comparitative studies — the scientific field that studies the relationships between legal systems — is largely focused on private law and in particular contract law, trust law, and anything else which practically applies to international trade.

4.1 Case texts

Differences in law between countries are not only confined to the notion of what the rules are and what the sanctions are when people break the rules. They can also be the result of a fundamentally different approach to the administration of justice. One such difference is the use of a jury — a group of randomly selected civilians — to give a verdict, but although this might make a difference for an individual case the impact on jurisprudence is negligible, because with only a few exceptions juries decide in criminal cases whether or not a suspect is guilty based on the evidence in the case. In other words, a jury interprets evidence and not the law and is therefore very unlikely to create any **precedent**[1] for that law. This is even more obvious when one considers that deliberation of the jury takes place behind closed doors and there is no argumentation given for their verdict. A decision without an accompanied reasoning can not serve as a precedent for future cases. The findings of the jury do not find their way to any written form, therefore the only difference as far as the legal texts are concerned will be a missing line of reasoning for the verdict.

A second — much more extensive — difference in the administration of law is that of common law versus civil law. Common law and civil law are the two main traditions within private law. Common law is within Europe practised in England, Wales and Ireland and was in the time of the British empire spread to all the countries across the globe that were at that time under British rule. It is founded on English customary unwritten law and is characterised by the absence of codification. Within the common law system every decision of the court is based upon the whole of previous decisions of that court, as opposed to civil law which has a systematic codification of its law. Civil law is largely based on Roman law en is practised in the rest of Europe. At first glance there seem to be large differences between common and civil law. In [22] J. Smits sums up seven points of comparison and after putting these into perspective concludes that despite their fundamentally different approach to law, their practice is far more alike then one would expect. When we take these points from Smits and look at their potential influence on legal text three of them remain and these will be discussed shortly below. For this purpose we will differentiate between effects made on the actual data presented in the text and the structure of the text. Data are the details of the case that are mentioned, like the facts of the event, statements of people, descriptions of circumstantial evidence, et cetera. Data can also be the motivation of the verdict, the verdict itself, or references to other cases. Structure then, is the way this data is presented to the

[1] In law, a precedent is a conclusion that is drawn in a case, which has an impact on the way similar cases will be handled. If the precedent becomes mandatory in future cases, it is called a *binding precedent*.

reader. The above mentioned jury system for instance is likely to only have an effect on data, namely the absence of argumentation for the verdict, and virtually none on structure. The importance of this distinction lies in the fact that different data will not introduce any significant difficulties in translation, whereas a different structure on the document level as well as on the sentence level is something to be reckoned with in dealing with the text, either by hand or by computer. The three points of difference with a possible influence on legal text are the logic of reasoning, precedence and culture.

Firstly the logic of reasoning, which is quite a difference between the two systems and closely related to the level of codification. In common law we see that reasoning takes place first en foremost based on the facts in the case possibly combined with precedence, whereas in civic law we find the logic of syllogism. Syllogism is a method of logical reasoning in which two statements, a so-called major premise and a minor premise, are presented, after which the conclusion results as a necessity. For instance:

Major premise: *When it rains, the streets get wet.*
Minor premise: *It is raining.*
Conclusion: *The streets are wet.*

In civic law the court follows this type of reasoning, whereby they start with a general judicial principle or rule (the major premise), then present the case (the minor premise) and finally come to the judgement (the conclusion) at the end as a logical result from the above. The approach in common law is thus a much more pragmatic one with a focus on the details of the case at hand, while in civic law the general principle is crucial and forms the basis of reasoning. The influence on the text of this difference is large in data and structure. In data because in common law a lot more factual details are given and in structure because the layout of the text as a whole is in accordance with the line of reasoning.

Secondly precedence, the fact that in common law precedence has such an important role is only reflected in a larger amount of references to previous cases, i.e. a difference in data.

And thirdly culture, which is really an umbrella term for the differences in writing style which influences the structure. In European common law especially the English court stands out with a style of writing that is quite elaborate and at times almost literary. This typical style is partly culture but also related to the attention for factual details and the typical reasoning and argumentation.

The differences in data and structure of case text between common and civil law are quite apparent and can clearly be traced back to the systematic differences of their respective systems. This is not to say that within the group of civil law countries case texts have an identical structure, also here we find differences in the extensiveness of the argumentation and the type and amount of references to outside sources like literature or even foreign law. There is one more dimension in which differences occur and that is time. Although courts in general have relatively conservative leanings, things do change over time in administrating law and case texts will reflect this. Perhaps on the long run the influence of the European Court of Justice on national administration of law will smoothen out these differences in structure, but for the time being it is more likely to add to the diversity already present.

4.2 Legislative texts

There are only a few occasions on which legislative laws have met. Firstly, there are the examples of the transplantation of an entire system of law into another country, as was done

with Dutch law in Indonesia starting in the beginning of the 17th century [13] whereby Dutch law was superimposed upon local customs. Another example would be Turkey in 1927 when the entire law system was renewed under Atatürk and they almost literally copied the codification from Swiss law[2]. In these cases there will be a clash between the local culture and the new legislation, but since the entire legislation is replaced and the receiving country also gets an entire new lexicon with it, difficulties with translation do not play a very big role in the transition. A more gradual transition took place — and still is taking place — in Scotland which started when they assimilated into the United Kingdom in 1707 and their legal system blended with that of the English. In the treaty leading to the union it was determined that Scotland would keep its own private law, therefore the whole really is a co-existence of two systems. Finally, there are those countries that have more than one official language and that practise the same law in those languages. European examples of this type of mixture are Belgium, Finland and Switzerland.

The European Union is at this point in time not unlike the Scottish example in the sense that part of the legal system is being replaced, but part of it remains in tact, the EU is unique however when it comes to its lingual diversity involving its 20 official languages.

5 Legal language

Judicial language is quite the typical lingo and although in layman's terms one will find it more than often referred to with terms like 'gobbledygook' or 'mumbo jumbo' an equally appropriate but more scientific term for the legal language would be a **register** [6]. Although most scientists actually refrain from putting a term on it and will mostly describe it as a part of language that not only has its own lexicon but also its own structure and style. all these characteristics are associated not primarily with a group of users — as does a dialect — but with their field of use: law. Every legal system has its own legal language and because of its unique properties it is quite a different thing to translate from English to Spanish, then it is from legal English to legal Spanish. In fact, it is an expertise of its own that requires a knowledge of English and Spanish, but also an extensive knowledge of their respective laws.

In discussing thesauri in Chapter 2.1 four different word-to-word relationships were defined: synonymy, hyponymy, homonymy and polysemy. Again, when translating legal text we run into difficulties whenever one of these relations is broken, missing or when two words appear to be related but are actually not, the so-called faux-amis. An excellent example is that of the difference between the two pairs of words for murder in Dutch and German: 'moord' & 'doodslag' and 'Mort' & 'Totschlag'. In general use these two pairs of words are perfect translation of each other, in a judicial sense however there is a significant difference between the two. Dutch law states in 289 Sr. that:

> *"Hij die opzettelijk en met voorbedachten rade een ander van het leven berooft, wordt, als schuldig aan moord, gestraft met levenslange gevangenisstraf of tijdelijke van ten hoogste twintig jaren of geldboete van de vijfde categorie."*

While German law states in StGB § 211:

> *"(2) Mörder ist, wer aus Mordlust, zur Befriedigung des Geschlechtstriebs, aus Habgier oder sonst aus niedrigen Beweggründen, heimtückisch oder grausam oder mit gemeingefährlichen Mitteln oder um eine andere Straftat zu ermöglichen oder zu verdecken, einen Menschen tötet."*

The important difference here is that in Dutch law the difference between 'moord' and 'doodslag' is based on whether or not the offender has planned the act in advance — "met voorbedachten rade" —, while in Germany the difference is based upon the severity of the motivation and context of the murder. This could very well mean that someone convicted for 'moord' in the Netherlands would in Germany be convicted for 'Totschlag' or the other way around. What appear to be synonyms are actually not.

The above example is just one of many, and a simple one to boot, for instance an attempt to compare the Dutch and German terms for 'legal act' gives us multiple terms in both languages that can only in some cases be considered equivalents[1]. This type of difference in concepts also arises in conventional translation when cultural differences influence the concept that lies behind the terms in both languages. In judicial translation however, because of its closed systems, terms less often refer to universal 'human' concepts

[1] For a more detailed description of this example see Pintens [18]

and the concepts are therefore less corresponding across different judicial systems [26]. Translating the word 'table' into Dutch ('tafel') is a relative easy task provided one has the knowledge or at least a dictionary because the English and the Dutch share the same concept of a table, namely a top with for legs. In law however, we often deal with unique or at least ambiguous concepts like for instance the English 'trust'[1], which is a typical construction that has no Dutch counterpart. This is in effect a lexical gap and legal texts are full of them. All these lexical gaps and other issues with equivalence often present so much of a problem that a translator is forced to resort to either of three alternatives [9]:

- *just leaving the term in its original form*, which is really only a realistic option if one can count on the intended audience to already know the meaning of the term. But it would be for instance possible for the term 'trust' as it is a well-known construction in international trade.
- *giving a description of the original term in the target language*, which is probably the most helpful method for readers, although it might annoyingly disrupt the continuity of the text flow.
- *introducing some neologism accompanied by an explanation*, which is of course only effective if this neologism gets picked up and used by other translators and readers alike. All in all this is a precarious and not undisputed method, because of the sometimes arbitrary choice of a neologism, the difficulties in spreading its use and because until it is widely accepted it still leaves the need for a description of its meaning.

The last alternative might receive some fierce criticism, but it does illustrate one important point and that is that in the long term the ultimate goal is to provide a translation that is besides accurate also constant over time and over all translators, irrespective of the chosen solution. With this objective in mind, access to a large collection of previous translations is an indispensable asset to a translator. The ability to effectively search through the large collection of national and international legal documents out there is a task that can be greatly improved by the Eurovoc project.

[1] A legal construction whereby an object is owned by one party (the trustee) and can be made profitable by another party (the beneficiary).

6 Automatic Indexing with Eurovoc

6.1 Descriptors and associates

The purpose of the system developed by the JRC is to automatically assign terms from the Eurovoc thesaurus to a document — this is called **indexing** — in any of the languages present in the Eurovoc project. At his point this indexing task is done by human experts, who themselves examine the document and then assign one or more terms from the Eurovoc thesaurus to it. One large difficulty in assigning Eurovoc-terms to documents automatically is that the terms, officially called descriptors, often consist of multiple words and therefore most of them have a relatively slim chance of actually appearing in the text. If a descriptor is assigned to a document by a human indexer, there is actually only a 31% chance that this descriptor is literally present in the text itself. On top of that, if a Eurovoc term does appear somewhere in the text literally, a human indexer would assign it to the document as a descriptor only 1 out of 10 times. Examples of some of the longer descriptors that are unlikely to be found in texts are: 'objections to an election result' or 'exploitation of the seas'. To account for this, one extra list of terms is added to the process, a list of so-called **associates**.

Associates are the terms that actually do show up in the texts and serve as an indicator for the descriptors, much like a flock of seagulls would be an indicator for the presence of fish under the water surface. Each descriptor has a long list of associates attached to it that are considered good indicators of the subject. For example the descriptor 'fishery' would get the associates 'fish', 'boat' and 'sea'. If the latter three words are found in a text then it is safe to assume that the text is about fishing and we can assign the descriptor 'fishery' to it. For more accuracy each associate has a certain **weight**, which is a number that is assigned to it that signifies how accurate the indicator is, the higher this weight is the more accurate it is in indicating a certain topic, i.e. the stronger it points to a certain descriptor. In the example we could imagine that 'fish' is very likely to appear in a text about fishery — it could also be in a recipe for a fish stew, but within the context of legislative documents this is not likely to be the case —. The word 'sea' on the other hand can appear in texts about fishery, but also in texts about the environment, tourism, transport, etc. In this case, the associate 'fish' would get a higher weight then 'sea' for this descriptor, note that for other descriptors this could be the other way round. Weights are discussed in more detail in section 6.3.

6.2 The basics of the system

Before the system is able to assign descriptors to a text it has never seen before it must have a list of descriptors and their associates with weights. This list has to be generated for every language for which the system is planned to be used. Generating this list is called 'training' the system and has to be done once only, as long as the thesaurus in use is not being changed. After the training it can be used to assign descriptors to any new text we might present it.

A simplified version of the whole system is depicted in Figure 2. The training process starts with a training **corpus**, which is a collection of documents that have descriptors — from the Eurovoc thesaurus — assigned to them manually by experts. These hand marked

documents ① are fed into the computer and from this the computer produces the descriptor-associates list ②. In this step all the documents that have gotten a descriptor 'fishery' are collected and scanned for the words they have in common and are used a lot. In the example these words called associates are 'fish, boat, sea'. As soon as this is done for all available descriptors the training phase is over and we are left with one huge descriptor-associate list, this list contains for each descriptor a long list of its associates. Next, in the running phase we feed a document ③ into the computer which has no descriptors, in fact we know nothing about it except its language. Based on the descriptor-associates list for that specific language we are now able to assign a descriptor to the document by reversing the process. We look for the most frequent words in the document and find that these are 'fish, boat, sea'. After going through the associate list for all descriptors we find that the descriptor 'fishery' has a list of associates that best match the words that are found in the document and thus the document gets the descriptor 'fishery'. It is normal for a document to get a couple of descriptors like this.

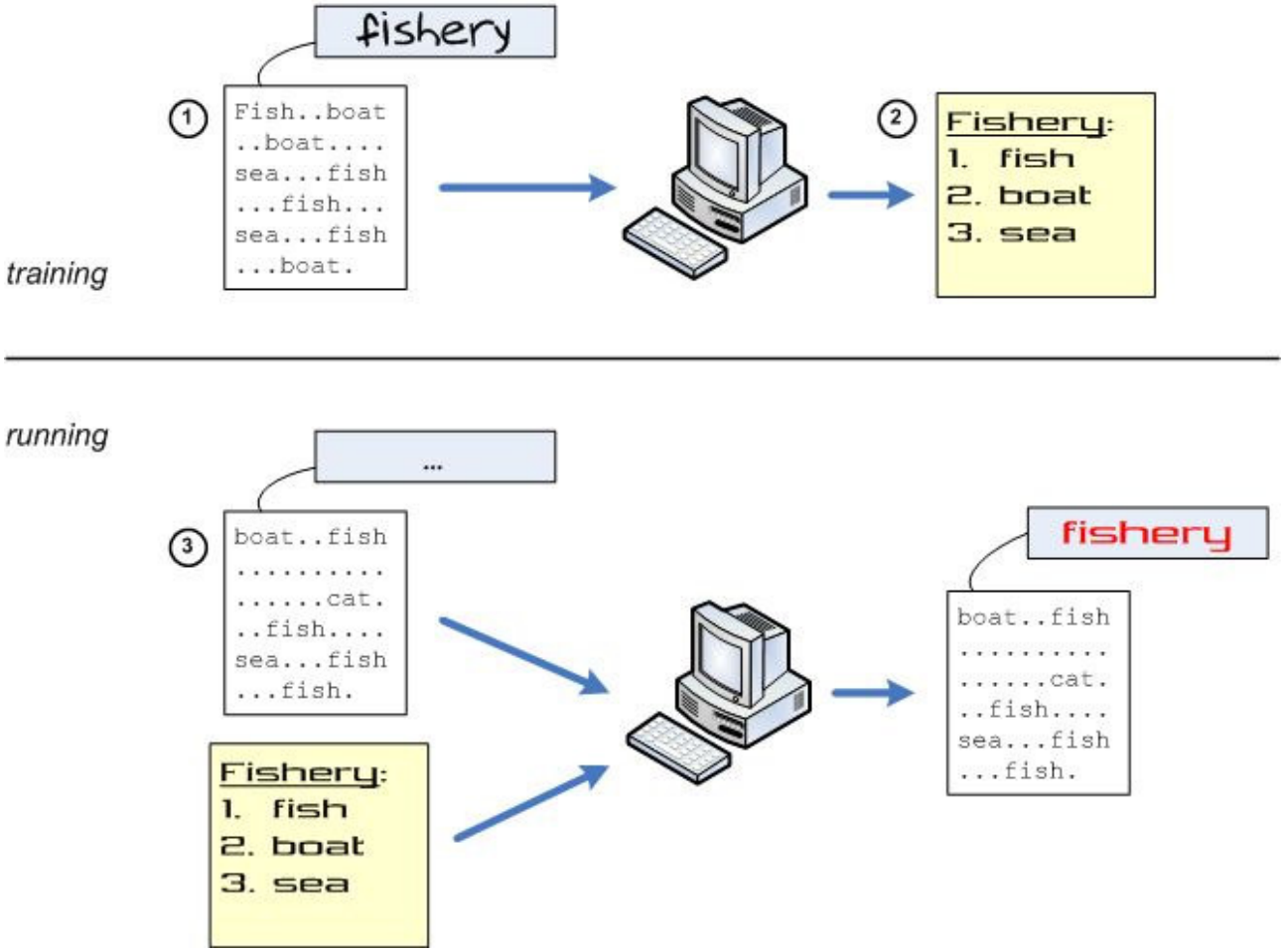


Figure 2: Indexing with Eurovoc

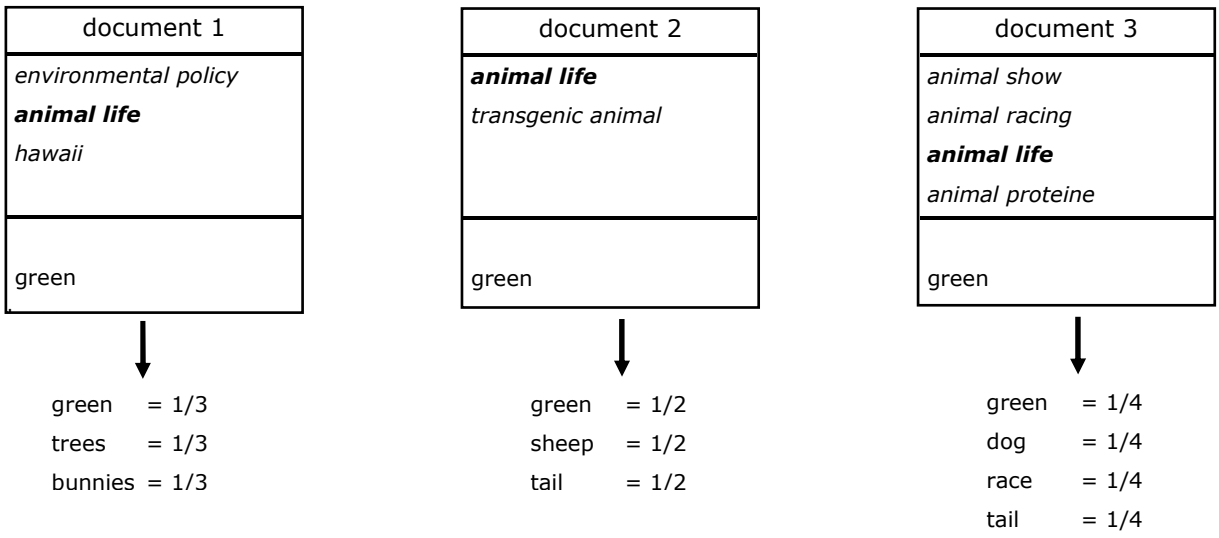
6.3 Training the system

In reality, of course there is a bit more to it. To start with, every document can have more then one label, but this fact does not introduce any new difficulties which need to be solved, in fact the same corpus gives us more training data per descriptor if we allow multiple

descriptors being assigned to a document. The whole process being described in this section and in the following one can be found in a more formal notation in section 9.3.

The associates for a single descriptor are selected as follows: for each unique word in the document, the log-likelihood ratio is computed and any word with a ratio above 0.15 is added to the list of associates. The log-likelihood ratio is a way of comparing the occurrence of a word between two corpora. In this case we want to compare the occurrence of a word in one document to the occurrence of the same word in our whole training corpus. For instance, if the word 'policy' appears often in the whole corpus, say on average twice in every document, and we count the occurrences of the word 'policy' in one single document and find that it occurs two times, this is no big surprise and we would say that an occurrence of two is nothing special, nor is three for that matter. But if we would count the word 'policy' and find it 7 times this would be surprisingly often and we would say that the word 'policy' must be really important in this one document and that it is thus an important associate. The exact computation of the log-likelihood ratio can be found in equation (11) on page 43, it suffices to say here that the greater the log-likelihood ratio is for a word, the more special a word is for the document. Anything with a ratio below 0.15 is discarded as unimportant and not considered as an associate. The threshold of 0.15 is selected by the JRC by trail-and-error and has shown to produce the best results. This log-likelihood ratio is only used for the initial removal of the very worst associates.

We know now which associates are important for which document, but what we really need to know is which associates are important for which descriptor and how important exactly. In other words, we need to go from our set of associates per document to an ordered list of associates per descriptor. We know for each document in our training corpus which manual descriptors it has, so to match associates to descriptors we can assign the associates of a document to the descriptors of that document. For instance, if a document has the associates 'green,trees,bunnies' and the same document has the manual descriptors *environmental policy* and *animal life*, then both these descriptors get 'green,trees,bunnies' as their associates. Exactly *how* important these associates are for a descriptor is determined by the number of times it appears for a descriptor, each time divided by the number of co-descriptors in a document. How this works for the descriptor *animal life* can be seen in the figure on the next page.



The final descriptor-associate list for the descriptor ***animal life*** now becomes in order of importance:

green = 1/3 + 1/2 + 1/4 = **13/12**
tail = 1/2 + 1/4 = **1/2**
trees = **1/3**
bunnies = **1/3**
dog = **1/4**
race = **1/4**

Figure 3: creating a descriptor-associate list

The number behind each associate — 13/12 for green, 1/2 for tail — is a measurement of its importance as an associate and is called the associate's **weight**.

In the example in Figure 1 we see that 'green' scores high as an associate for *animal life* because it appears in a lot of documents that have this descriptor. There are however associates that are so very common that they would almost appear in every document. For example a word like 'policy' will appear quite often in legislative documents and will therefore end up getting a large weight, which makes it an important associate, but at the same time it will be an important associate for a whole lot of descriptors. If the word 'policy' is an important associate for many descriptors we still don't know which descriptor to choose from all of them if we spot the associate 'policy', so the weights for these kind of words need to be lowered to prevent the situation where all descriptors have the same top 20 of associates with common words like 'policy' or 'regulation'[1]. This is done by getting the frequency of the most common associate in the whole corpus and then lowering the weight for every associate with a corpus frequency of at least 10% of that. Furthermore, associates with a very low weight are cut of the list.

[1] The very worst of these common words are already filtered out by the use of a stop word list, but there are still enough left to be a disturbance.

6.4 Running the system

With this descriptor-associates list we are able to feed an unknown document into the system and get the descriptors for it. For each descriptor we have a list of associates and their weight, and for the new document we can construct a similar list with weights for the associates that it contains. Selecting descriptors for a document is now a matter of comparing each of the descriptor-associate lists with the list for the new document and select the most similar.

These two lists can also be seen as n-dimensional vectors, with n being all possible associates and the components of the vector being the weight of these associates with a zero if the associate does not occur the list. This vector space model is commonly used in the field of information retrieval and provides us with some methods to check the similarity between two lists of weighted terms. There are several well-known methods for comparing vectors, of which the simplest are the scalar product and the cosine method. The JRC also uses the okapi formula [21] which is not based on the vector model but instead uses a more elaborate method of comparison that also includes the length of associate lists and the total number of descriptors. The scalar product is the same as the inner product (a·b) of two vectors a and b, the formula for the cosine method can be found in equation (20) on page 44, and the okapi formula can be found in Appendix I. Depending on the language, a combination is chosen between the two vector based methods and the okapi formula, but for testing purposes the JRC often only uses the cosine method and so have I in my tests. The descriptors that best matches with the new document according to the vector comparison are likely to be appropriate for the document. So we actually end up with a long list of all available descriptors and a number which indicates how well they match the text, let's call this number the similarity factor.

For the practical use of assigning descriptors to documents, we could for instance take the best three descriptors and assign these to the document or perhaps assign all descriptors with a similarity factor above a certain minimum — perhaps with a certain minimal and maximal number of descriptors to assign — and then discard the rest of the descriptors on the list.

6.5 The results

The above described method has been developed and tested at the Language Technology department of the JRC and the results show an average **recall** of about 50% and a **precision** of about 45% for all languages when compared to assigning descriptors by hand, at a rank of 5 descriptors. This means that when we consider the best 5 computer-retrieved descriptors for every document, that on average 50% of the manually assigned descriptors will be among these 5 and that 45% of these computer-retrieved descriptors will be accurate.

The results can also be depicted in terms of an F-measure, which combines the recall and precision in one single number with the possibility to put more emphasis on either recall or precision. The F-measure is calculated as follows:

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} \tag{1}$$

By changing the value of α in a range from 0 to 1 we can give more weight to recall or precision. Recall and precision is all about the size of the fishing net. If we go fishing for tuna, we can get a huge net and catch 12 tuna fish, but also 3 dolphins, a turtle, 2 anchors and a ton of seaweed. That's high recall — 12 tuna fish — but low precision, because we're stuck with a load of unnecessary stuff. If, on the other hand, we go fishing with a fishing rod and some tuna-bait we might catch 4 tuna fish on a day. That's low recall but some pretty high precision. In the case of descriptors, if we would prefer recall over precision, it means that when the computer assigns a list of descriptors to a document, we place great value on getting at least all the right descriptors in that list and care less about the fact that there might also be some wrong ones among them. In this case we would take a small α. In the other case, when we prefer precision over recall, it means that we do not care as much to get all of the right descriptors, but we care about the list of descriptors returned by the computer to contain as few wrong ones as possible. Then we would get a large α.

The JRC places equal value on both and measures its results with an even distribution between recall and precision, in that case the formula can be simplified to the following: $F = \frac{2PR}{P+R}$. With the average results for all languages mentioned above, we would get an F-measure of about 0.47. The F-measure does not only give us the possibility to express the relative importance of recall and precision, but also the convenience of a single number to evaluate results.

There is one important thing to be noted when considering the quality of this measurement. It was said that for our test documents the descriptors that are assigned by the computer are compared to the descriptors that are assigned by a human expert. The results of this comparison are expressed by the recall, the precision and the F-measure. This implies that by these measurements the best possible result achieved by the computer is getting the exact same descriptors as the human expert. But how good is the human expert? To figure this out the JRC had a panel of other indexing experts evaluate the work of the manual assignment of the English and Spanish corpus. The panel approved of 74% of the manual descriptors for English and 84% for Spanish, which means the panel thought that the precision achieved by manual indexing was 74% for English and 84% for Spanish. In other words, there is no such thing as a perfect set of descriptors because any expert would do things slightly different, but if automatic assignment can come close to a precision of about 80% it would be as good as an average human indexer.

6.6 Improving the results

Apart from the fact that there are descriptor-associates lists for each language specifically, this method of the JRC to automatically assign descriptors to documents functions without treatment of the texts beforehand or any kind of language specific processing and has delivered some good results. This approach was taken on purpose to allow the system to be used independently of languages given its use in a highly international context. This does not mean however that linguistic processing cannot be useful in improving the results. Experiments with linguistic pre-processing for English, French and Spanish show a further improvement of the F-measure by some six to eight hundredth.

This thesis focuses on the question in which way the Lithuanian texts that serve as input for the computer can be prepared to improve the performance of the automatic indexing in effectiveness. Efficiency is not considered a priority as all of the analysis and indexing is done beforehand and not during the time of a user's query to a database. The

research concerns the pre-processing in the training phase as well as the indexing phase and is focussed on two areas: pre-processing at word level, looking at morphology, and pre-processing at document level, looking at the structure of the document.

6.7 Stop list

The first improvement to the automatic indexing for Lithuanian came from the JRC and consisted of the use of a **stop list** that was provided by the CCL, which has increased F-measure by almost 0.06 points to 0.48 (P/R at 45.9/51.4%). A stop list is a straightforward list of the most frequently used words in a language or a subset of a language. Stop lists are gathered from large corpora by counting words and gathering the most frequent occurring words in a list. In this case a list was used that was aimed at the specific genre of texts, namely Lithuanian legislative documents and was gathered from a part of the corpus from the KLC. The words in the list are so frequently used, that they are considered not discriminative enough and are skipped in parts of the process to prevent them from becoming statistical noise. All texts that serve as input for the system, in the training phase as well as in the indexing phase, are matched against this stop list and all stop words are ignored as associates.

7 The Lithuanian language

The Lithuanian language today is spoken by some 4 million people, of which 3 million in Lithuania itself — out of a 3.5 million population — and another million by emigrants in neighbouring countries and Lithuanian communities all over the world. The Lithuanian alphabet consists of the normal Roman letters, with the exception of the letters q, w and x. In addition to this there are ten extra letters formed by accented Roman letters, which makes a total of 32 letters.

12 vowels
a ą e ę ė i į y o u ų ū
20 consonants
b c č d f g h j k l m n p r s š t v z ž

Table 1: The Lithuanian alphabet

televizorius	television
paveikslas	picture
šakutė	fork
kėdė	chair
stalas	table
šuo	dog
kiaušinis	egg
bananas	banana
ananasas	ananas

Table 2: Some examples of Lithuanian words

The Lithuanian language is part of the large group of Indo-European languages. The group of Indo-European languages include most of the European and Western Asian languages, which is an estimated 3 billion people speaking some 443 languages. Lithuanian is one of the two living languages in the subgroup of Baltic languages, the other one being Latvian. Lithuanian is very much an archaic language in the sense that it shows a lot of features that used to be part of old languages, but that most modern languages have lost. To understand the importance of this fact we have to go to the bronze age, to the origin of the Indo-European languages. The language from which all Indo-European languages are thought to originate is called the Proto-Indo-European language and it is a hypothetical language. It is a hypothetical language because there are no actual authentic sources for this language and it is constructed based on what scientists *think* the Indo-European people might have spoken back in their early bronze age. This hypothetical original language is constructed based on comparative studies being done on the Indo-European languages we do have sources for and although it is not undisputed where exactly these people originate from or how their language has spread, there is a general agreement on what the language itself must have been like. Lithuanian is believed to be the closest of all modern languages to the Proto-Indo-European language. This fact and the fact that it is still alive, i.e. still in use in written word and speech, makes it an invaluable source for linguistic studies. There are quite a few of these archaic features in the Lithuanian language, such as its lexicon, sound, conjugation of

verbs, word order, and many more[15]. One of these remarkable features that sets it aside from most modern languages is the rich inflection of nouns and this is a feature that is in fact interesting to us here, as is has an impact on the statistics that we use in the automatic indexing.

7.1 Declension

Inflection is a change in the form of a word to indicate a change in its grammatical function. When this is done with nouns, pronouns and adjectives, which is what we are talking about here, it is called **declension**. In Dutch as well as English declension has disappeared for nouns, but in German for instance there is still some left as we can see from the following example:

1. Der Apfel ist grün. (The apple is green.)
2. Die Farbe des Apfels ist grün (The colour of the apple is green.)

In sentence 1 the word 'Apfel' is the subject of the sentence and stands in the nominative case. In sentence 2 the colour belongs to the apple, so the article and the word 'Apfel' itself stand in the genitive case, in genitive 'der' becomes 'des' and 'Apfel' becomes 'Apfels'. In English this kind of possession would be indicated by using a preposition like this: *The colour **of** the apple is green,* while the actual noun would remain unchanged.

The system of declension for Lithuanian is an extensive one, with seven cases for the singular — one object — and another seven for the plural — more than one object, so a simple word like table can thus take on fourteen different forms. The language used to also have a dual declension which was used when there were exactly two objects, but this has disappeared in everyday use.

	Singular	Plural
Nominative	stalas	stalai
Genitive	stalo	stalų
Dative	stalui	stalams
Accusative	stalą	stalus
Instrumental	stalu	stalais
Locative	stale	staluose
Vocative	stale	stalai

Table 3: Declension of the word stalas (table)

In relation to the automatic indexing, one could see how it would be hard to extract a single term from a text if it can potentially be written in 14 different ways. When we are counting the frequency of words in the text we might end up with 14 different numbers for 14 different words, even though every single one of these words represents the concept of 'table'. However the system is not aware of this and they are all included as being separate associates, thereby messing up the statistics. What we really want to have is a single number showing us how often the concept 'table' is present in the text.

Experiments from the JRC with English, French and Spanish have shown that the pre-processing of texts with a linguistic perspective and aimed at one specific language can

improve the results. There is no reason to believe that comparable results could not be made with the Lithuanian language, in fact the highly inflectional nature of the language would predict the 'basic' results — results without any pre-processing — to be lower and therefore to offer a higher potential gain for pre-processing. All because inflection introduces complexity to the text that obscures our statistics.

We find support for this reasoning in the research of Willett and Popovič [19] who researched the effect of **stemming** in Slovene texts — Slovene has 10 distinctive inflectional forms. Although their research was aimed at a natural language query and made no use of a thesaurus, both their research into querying and the matter of composing a Eurovoc associate list consists of keyword extraction and are therefore comparable procedures as far as the use of stemming is concerned. Willett and Popovič found that without stemming the use of Slovene texts produced worse results than the use of English texts and that at the same time, stemming meant a much bigger improvement for Slovene texts. They concluded:

> *... the effectiveness of a stemming algorithm is determined by the morphological complexity of the language that it is designed to process.*

The same effect can be seen again with the JRC results for automatic indexing for Eurovoc, as we see that their test results for Lithuanian give an F-measure of about 0.425[1] (Precision/Recall at 44.0/41.2%), which means Lithuanian is lower then the average for other languages, and this could very well be due to inflection.

7.2 Stemming

If we could bring all the inflected words back to one basic form, i.e. stemming, we might be able to improve the results. Most stemming tools are based on variations of the well-known Porter stemmer [3]. In 1980 Porter built a stemmer for English that worked on the basis of a list of known suffixes and some rule set to determine if and how to strip a word from its grammatical morphemes[2]. The result of this procedure is that a word gets truncated to a shorter version, a so-called 'stem'. A Porter-like stemmer for only the nouns and adjectives in Lithuanian could very well be beneficial to the automatic indexing process. In the past, good results have been made with stripping inflectional suffixes — see chapter 3.2 — and nouns and adjectives are in general more contentious then verbs. By the latter is meant that nouns and adjectives will tell more about the contents of a text then verbs, this seems not only intuitively right, but was also shown by Kraaij and Pohlman [4]. In their research into the results of stemming on information retrieval they looked at the syntactic category of the best query terms, i.e. the terms that retrieved the highest number of relevant results. What Kraaij and Pohlman observed was that among the best query terms there were 58% nouns , 13% adjectives and 29% verbs and if restricted to the single best query term for each query these percentages were even 84%, 8% and 8% respectively. It should be noted that because of the use of a stop word list the most obvious group of verbs without meaning with respect to content, the auxiliaries, are already removed and are not included in these percentages. What this means is, that when nouns and adjectives are used to describe the

[1] F-measures and Recall/Precision percentages in this paragraph are all measured at rank 5, i.e. only the first 5 returned descriptors are taken into account.

[2] A morpheme is the smallest part of a word that still carries meaning. e.g. 'birds' has two morphemes: 'bird' and 's', the first has an obvious meaning and the second adds the plural, it means there is more than one bird.

contents of a text in a search query, the results of the retrieval are far better than when we would use verbs, in other words their results show that nouns and adjectives together seem far more meaningful when we want to describe the content of a text than verbs.

A general recognition of the meaning of nouns and adjectives, the relative low results for Lithuanian documents in the JRC's automatic indexing and the fact that for Lithuanian stemming nouns and adjectives is actually less complicated then stemming verbs, seemed enough reasons to start off with a focus on stemming nouns.

At this point the question arises how grammatically refined the stemmer has to be. Do we have to use a stemmer that will be able to do a proper morphological analysis of a word, resulting in a grammatically correct separation of a words stem and suffix or will a more superficial method suffice, in which words are truncated by the computer without a grammatical awareness, possibly resulting in grammatically incorrect stemming? A study into this very question concerning the Finish language — Finish is also a language with a very rich morphology — seems to favour the second. In [1] the authors seek an explanation for *"[the] fact that even the crudest truncation procedure .. yields a surprisingly high degree of correctly segmented word forms"*. In other words, what is the reason that when we do not consider grammar rules and simply chop off, i.e. truncate, an ending when we think it *might* be a suffix we hardly ever end up with a stem that is too short and an ending that is too long? The reason for this, according to the authors, can be found in the **phonological** differences between a stem and a suffix. In Lithuanian for instance all 1-letter noun suffixes are vowels and with 2-letter suffixes an 'n' or an 's' is introduced in half of the cases, further — the suffixes reach on to a length of 6 — only r's, m's and some j's are introduced. The combination of these five consonants and all the vowels lead to a selection of suffixes that differ from phonemes near the end of stems. Given just a single phoneme a human being should not have much trouble differentiating between a suffix and the end of a stem. One hypothesis is that this phenomenon develops over time, because being able to *hear* a difference between parts of a word that actually have a different grammatical meaning helps us in communication. Because they don't sound alike and therefore don't look alike, a computer is less likely to mistakenly recognise the end of a stem as a suffix or part of a suffix.

7.3 Derivational suffixes

Derivation is the process where we add something to a word, which is not a word in itself, to form a new word with a new meaning. Two types of derivation are worth mentioning here. The first is the diminutive, whereby we add a suffix to a noun when we want to make clear it is something small. The English language does not have diminutive suffixes, but in Dutch for instance we can use some form of '*-je*' as in: kop*je*, zon*netje*, water*tje*. Although Lithuanian knows about 80 different diminutive suffixes, there are actually only four different suffixes that are commonly used for diminutives, they are *elis*/*elė*, *ukas*/*ukė*, *utis*/*utė*, and *ytis*/*ytė*. For each form there is thus a masculine ending with *–s* and a feminine ending with *–ė*. For instance, the word for 'table' is 'stalas' and the word for 'little table' — in Dutch 'tafeltje' — would become 'stalelis'. In theory it is even possible to combine more than one of these suffixes if one would want to exaggerate it a bit, so a combination of the suffixes *elis* and *ukas* would make a extremely tiny table 'staleliukas'. These diminutive forms are also subject to inflection just like a 'normal' noun is and in fact follow the same declension patterns. The declension of the word stalelis can be found in Table 4.

	Singular	Plural
Nominative	stalelis	staleliai
Genitive	stalelio	stalelių
Dative	staleliui	staleliams
Accusative	stalelį	stalelius
Instrumental	staleliu	staleliais
Locative	stalelyje	staleliuose
Vocative	staleli	staleliai

Table 4: Declension of the word stalelis (little table)

In theory these extra diminutive suffixes could cause the same problem as inflectional suffixes do, namely increase the amount of word forms and thereby mess up the statistics used in the indexing process. To examine the impact diminutives could have, a spot check was performed on the test corpus (3 million words), whereby the following diminutive suffixes were examined: *elis* (nominative masculine), *elio* (genitive masculine), *ukas* (nominative masculine), and *uko* (genitive masculine). In Table 5 are the number of occurrences of these diminutives. Any words that might look like a diminutive but are actually not, are not counted, e.g. the word 'butelis', which means 'bottle', ends in *–elis* but is not a diminutive and is therefore not counted.

suffix	occurrences
-elis	94
-elio	419
-ukas	15
-uko	46

Table 5: Occurrence of diminutives in test corpus

Needless to say the numbers in Table 5 are so small on a test corpus of 3 million words, that diminutives will have a negligible impact on the indexing process and it is safe to leave them as they are and to only concentrate on the inflectional suffixes.

There is a second form of derivation worth mentioning, which is applicable to adjectives, namely gradation. Gradation is used with adjectives when we compare things. There are three degrees: positive (or standard), comparative, and superlative, Table 6 shows an English and Lithuanian example of adjective gradation.

	English	Lithuanian
standard	great	didis
comparative	greater	didesnis
superlative	greatest	didžiausias

Table 6: Lithuanian adjective gradation

Similar to the test for diminutives the impact of the suffixes for gradation was examined on the test corpus of 3 million words, resulting in Table 7.

suffix	occurrences
-esnis	4452 (516)
-iausias	1226 (400)

Table 7: Occurrence of adjective gradation in test corpus

As we can see in the table gradation is a little more frequent than the diminutives but still the occurrence is nothing to reckon with. Even more so when we account for the fact that there are two adjectives, didis (great) and mažis (small), that are mostly used and their meaning is quite abstract. With the latter is meant that they serve no purpose as an associate as they reveal nothing about the topic of a text. If we do not count these two adjectives when we examine our test corpus, we get the number between parenthesis in Table 7, which makes for an even less impressive number of occurrences. Therefore it was decided to also forgo on accounting for gradation suffixes when building a stemmer.

For all of the reasons above the first tool I have developed in the interest of this thesis was a simple suffix stripper that could reduce all different *inflectional* forms of a Lithuanian noun, pronoun or adjective to a common stem.

8 Blind noun stemmer

8.1 Stemming Algorithm

Because adjectives in Lithuanian are declined almost similar to nouns and it would therefore be hard to separately handle these two groups the developed stemmer uses a list of all noun and adjective suffixes, whereby the group of adjective suffixes only adds 5 'new' ones to the list of 105 noun suffixes. The algorithm used is quite simplistic. Ignoring all the words in the stop list, each word is checked to see if it ends in any of the known suffixes starting with the 6-letter suffixes and going down the list to the 1-letter suffixes. If a suffix matches, it is stripped from the word and the stemmer skips to the next word. Every word is thus stripped of an ending zero or one time(s). The stop words are left intact because we need to be able to recognize these later in the process.

8.2 Results after stemming

With this stemmer the original F-measure at rank 5 is increased by 0.023 to 0.505, with P/R at 45.8/ 56.3%. This means Precision is 0.1% less, but Recall 5% up; overall a good result. The results at other ranks show improvements in the same order of magnitude. When the stemmer has done its work we are left with a text with the stop words left untouched and all of the other words stripped of a suffix if one is found. This is quite a 'brute force' approach because it will strip appropriate suffixes but with quite some collateral damage while doing so.

Firstly, it is possible that a longer ending is stripped — because these are checked first — where the actual suffix is a shorter one. An example of this would be the word for woodpecker: genys. Let us take the example of a text in which this word would appear in the nominative, genitive and accusative singular: genys, genio and genį. The grammatically correct stem of this word would be ‘gen-‘, with the suffixes –ys, -io and –į, but since our list also happens to include the suffixes –enys and –enį and longer suffixes always precede shorter suffixes the stemmer will process these three forms as follows:

word	largest suffix found	resulting stem
genys	enys	g
genio	io	gen
genį	enį	g

Table 8: The largest suffix prevails

As can be seen in Table 8 the stemmer produces two stems for the same word, one being grammatically incorrect, but even in this rare[1] case we have been able to reduce the number of forms from three to two. The stemmer will in the worst-case scenario leave us with as many forms as we had before the stemming took place, but in most cases it will reduce the number of forms. The fact that these stems can be grammatically wrong is irrelevant, seeing as the system will merely proceed to count the words and the text does not have be readable

[1] See the discussion on phonological differences between stems and suffixes in section 7.2.

by any human in this process, whether 'g' stands for woodpecker or for submarine is all the same.

Secondly, *all* types of words are stripped, this means that also verbs, adverbs, etc. are truncated if they happen to have an ending that is in the list. Again, this simply replaces one variant with the other and does not add to the multiformity.

Thirdly, derivationally related nouns, adjectives and adverbs are most likely to produce the same stem. In the case where their meanings will also be closely related this can be considered a positive effect.

As was mentioned before the stemmer does not deteriorate our results because it will always reduce multiformity of words and never increase it. There is one small but to this however: it is theoretically possible that two semantically unrelated words get reduced to the same stem. If that would happen two words with different meaning start counting as one associate, which could have a bad effect on the quality of our associate lists. The fact however that two words have akin stems, will often also indicate a semantical relationship, so chances are not large of such a thing happening.

In short, there are two possible advantages of a morphologically correct approach to stemming as opposed to using the blind stemmer:

- We would have one stem for every word, whereas the blind stemmer every now and then produces two or more stems for the same word.
- Two different words would not get the same stem by chance, but only when this is morphologically correct, in which case they are more likely to be also semantically related.

Both of these, especially the second one, could lead to marginal improvements.

8.3 Lemuoklis

One of the tools provided by the KLC is MorfoLema[1] or in Lithuanian: Lemuoklis [27]. Lemuoklis is able to analyse a given word and determine its possible morphological structure, i.e. it is able to decompose a word into a stem and affixes and determine their properties. The word 'possible' is important in this definition of the program as one form of a word could have more than one decomposition. For example 'dažai' could be a noun, nominative, plural, meaning 'paint' or it could be a verb, second person, singular, present tense, meaning 'you paint'. Both would get the stem 'daž-'.

The use of Lemuoklis could give us a more morphologically correct stemming procedure. In order to determine the possible use of Lemuoklis a test was set up in which Lemuoklis analysed a part of the training corpus. The total amount of words it analysed during this test was about 5.6 million. It is important to note that these are not 5.6 million unique words. The goal of the test was to examine the performance on a realistic basis and by repeating the analysis and counting its results every time a word is repeated in the corpus gives us a more realistic figure. If it performs badly on common words this will have a big impact on its practical use and by running it on the corpus and not on a dictionary this impact will be reflected in the result of the test. Of the 5.6 million words 29% was unrecognized, meaning that Lemuoklis was unable to determine their morphological composition. The amount of words that was recognized and determined as some word with

[1] A demonstration program is available at: http://donelaitis.vdu.lt/~vytas/lemo/angl/lemo_down.htm

inflection — these are the nouns[1], proper nouns and adjectives — sums up to about 22%. However, from the last group only 48% could be reduced to one single morphological decomposition, the other half had two or more possible decompositions. Added to this is the amount of inflected words in the 'unrecognized' group and the fact that Lemuoklis has a certain error rate of its own, which leads to the assumption that a positive morphologically correct stemming of inflected words with the aid of Lemuoklis is possible for about 40% of all inflected words, which amounts to about 11% of all the words in the corpus. Considering this relatively low percentage and a projected small result in terms of improving the indexing process it was determined not to incorporate Lemuoklis in any further testing.

All in all we can conclude from the tests with stemming done here, that for the Lithuanian language stemming of inflectional suffixes does indeed lead to significant improvements for a statistical analysis task like automatic indexing for Eurovoc. Furthermore, that the method of stemming does not need to be very sophisticated and does not necessarily have to result to grammatically correct stems to see improvements. This leads to the assumption that similar rudimentary suffix strippers could also be beneficial to other languages with a rich inflectional morphology.

[1] Lemuoklis makes a distinction between nouns and numbers written out in full, both are called nouns here.

9 Semi-manual segmentation

9.1 Discourse segmentation

After the focus on morphology of words, this and the following chapter will take a broader look at the possibilities of exploiting the knowledge of the structure of a whole document to improve the automatic indexing. To this end the possibilities of automatically subdividing a document into its structural segments were examined, and if and how this technique could be used to enhance the automatic indexing.

Any well-written text is subdivided into segments, e.g. chapters, sections, paragraphs, etc. and knowledge of these segments can help in selecting the right keywords for a text. For instance, a term that is mentioned in sections like abstracts, summaries, or conclusions is likely to be more descriptive than a term in a random paragraph, which in turn is probably more descriptive than terms that occur in appendices.

Most document collections that have a connection with Eurovoc are of a legislative nature — parliaments and governmental organisations are the main users — and are strictly codified, i.e. the documents have a specific template with a lot of subsections, each with a specific heading. This means that these documents can be easily segmented if we have knowledge of the templates. There are, however, also documents that do not conform to any template or if they do it is not obvious from their lay-out. Some of these texts appear as appendix to codified documents.

9.2 Document structure

The method of segmentation used for this research consists of two steps. The first is to compose a list by hand of all the different types of sections that can be found in the texts and a way to identify them. With this list we can have the computer identify and single out those sections so we are able to process them differently from the rest of the text. The actual segmentation is thus done automatically, but it is based on manually gathered information on the segments. Fortunately, most of these legislative texts are quite formal while having a uniform layout and this is also true for our Lithuanian corpus. A review of the different types of document available in the Lithuanian corpus reveals that most of the codified documents contain the following elements: Title, Abstract, Preamble, Operative, Closing and Appendices. The order in which these appear is always the same, but apart from the title and the closing none of these sections are mandatory. The preamble and operative can be further divided into clauses and then into articles, but also this is optional. An appendix can contain anything. An XML-schema of this structure can be found on the next page.

```
<?xml version="1.0"?> <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema">

<complexType name=”document”>
<element name=”title” type=”string” />
<complexType name=”text”>
  <element name=”abstract” minOccurs=”0” />
  <complexType name=”preambule” minOccurs=”0”>
    <element name=”preClause” maxOccurs=”unbounded” type=”string”/>
  </complexType>
  <complexType name=”operative” minOccurs=”0” mixed=”true”>
    <complexType name=”opClause” minOccurs=”0” maxOccurs=”unbounded”
mixed=”true”>
      <element name=”article” minOccurs=”0” maxOccurs=”unbounded”
type=”string”/>
    </complexType>
  </complexType>
  <element name=”closing” type=”string” />
  <complexType name=”appendix” minOccurs=”0” maxOccurs=”unbounded”
mixed=”true”>
    <element ref=”text” minOccurs=”0” />
  </complexType>
</complexType>
</complexType>

</schema>
```

Figure 4: XML-schema of the general document structure

Because our corpus consists of plain text documents it is not possible to single out captions of figures, tables of data or any other parts of the text that in a document with a rich mark-up — like a Word file or a PDF — could be identified by a deviant font-size, font-weight, alignment or other mark-up of that kind.

Therefore the only way to identify the start of a section is to look for a number of key phrases, i.e. phrases that always appear at the start or end of a section. For instance, in a resolution there is a section in which the operative clauses — these are the actual decisions or actions that a resolution proposes — are placed and this part of the text usually starts with a phrase like “priėmė šį direktivą”, which means something along the lines of “passed this directive”. Whenever we find this phrase in the text it almost certainly marks the start of the operative clauses. Similar key phrases exist for chapter headings, article headings, footers and appendices. For the purpose of collecting these key phrases I have built a software tool that runs through the corpus and identifies key phrases based on regular expressions. Because available engines for working with regular expressions offer far more functionality then is necessary here and they do this at the cost of efficiency, a small and simple engine was constructed just for this purpose that can handle regular expressions containing characters, numbers, white spaces and line breaks in a certain range of cardinalities.

The list of the most common key phrases was accumulated by means of an inductive trial-and-error procedure, or rather a trial-and-success procedure. This procedure was repeated for every type of key phrase — types are *chapter heading*, *operative, footer, etc.* — and was done by picking key phrases out by hand. For instance, when collecting the key phrases that indicate the start of the operative clause, one would simply look at the document, find the operative clause and write down the phrase that announces the start of the operative clause. Because there are a number of these key phrases and 9749 documents to go through, this process was somewhat automated. The first key phrase from the very first document in the corpus was added to the — thus far blank — list and this list together with the entire training corpus served as input for a software tool that returned a new version of the corpus with the exception of all the documents that contained the key phrase in our list. From the newly derived corpus a new key phrase is picked by hand, added to the list and the whole process is repeated until all of the most common phrases are on the list. The collected list does not entirely cover the whole of the corpus because of documents that do not conform the general structure or simply because of typo's or exceptional spelling or styling of key phrases. With the collected phrases however about 91.4% of the corpus is covered and this percentage can thus be segmented based on key phrases.

9.3 The indexing algorithm including segment weighting

As explained in chapter 6 there are two major phases to the Eurovoc system, a training phase and an actual indexing phase, where the latter is either for testing purposes or for actual real-life application. In both of these phases statistics are gathered on the frequency of every word in the documents, in the first phase on a training corpus and in the second phase on a test corpus or a document that is submitted by a user, in all cases these frequency statistics are gathered in an early stage of the procedure. In this stage of the process the context of the individual words is still known and we are able to discriminate on it, i.e. give a word a different weight if it is in a certain segment. Because these weighting adjustments take place in an early stage in every phase it was necessary to reproduce the Eurovoc system in its entirety and make the necessary adjustments for testing segment weighting. Below is the entire algorithm as it was used in this project.

The procedure from chapter 6 stems from the JRC and is described by the original authors in [25]. What can be found below is the same procedure, adapted to include the possibility of term weighting and written down in a more formalised manner for an improved insight in the details of the procedure.

Training phase

First we build the weighted frequency lists for individual documents ($docFreq$) and the whole corpus ($corpFreq$). A basic non-weighted frequency list would consist of the raw frequency of a word in a document, e.g. if the word *stalas* would occur 50 times in a document called "*direktiva apie medininių stalų*"[1] then $docFreq_{x,y}$ would be **50**, where x is the word stalas and y is the document called "*direktiva apie medininių stalų*". This number 50 indirectly signifies the importance of the word *stalas* for this specific document. The actual changes to the JRC procedure take place in this phase and deal with altering these frequencies of words in documents, i.e. in the example we would change the number 50 to something else. We do

[1] "A directive on wooden tables."

this as follows: every time we normally see the word stalas in the document we count it as 1, instead we will now count it double, triple, or perhaps just half if we see it in a specific segment. Let us say the word stalas appears 50 times in the whole document, out of which 1 time in the title, 20 times in an appendix, and 29 times in the rest of the text. Because titles are important we double the weight of words in a title and because appendices are not we count them only half. The weighted frequency of the word *stalas* now becomes:
2 x 1 + ½ x 20 + 1 x 29 = **41**
The weighted frequency is lower than the raw frequency, because *stalas* occurs in the appendix a lot and appendices had a low weight. By giving different segments different weights in this way it is possible to alter the frequency of words that appear in a certain segment and thereby alter their importance as an associate for the document. Note that when a word is in the stop list its weighted frequency becomes 0, this is the stop list which is discussed in chapter 6.7.

$$stopwords = \text{list of stop words} \tag{2}$$

$$docs = \text{ the set of all documents} \tag{3}$$

$$descs = \text{ the set of all descriptors} \tag{4}$$

$$occurrences_{d,w} = \text{ the set of all occurrences of word } w \text{ in document } d \tag{5}$$

$$rawFreq_{d,w} = \left|\text{occurrences}_{\text{d,w}}\right|, \text{ the raw frequency of word } w \text{ in document } d \tag{6}$$

$$weightFreq_{d,w} = \sum_{o \in occurences_{d,w}} weight_{o,w}, \text{ the weighted frequency of } w \text{ in } d, \text{ where} \tag{7}$$

$$weight_{o,w} = \begin{cases} 0 & w \in stopwords \\ \kappa & o \text{ in title} \\ \lambda & o \text{ in preamble} \\ \mu & o \text{ in chapter heading} \\ \nu & o \text{ in appendix} \\ 1 & \text{else} \end{cases} \tag{8}$$

κ, λ, μ and ν are the weights we assign to the different segments.

When accumulated over all the documents the result is the weighted word frequency for the whole corpus (*corpFreq*). We will need this number later when we compute the log-likelihood.

$$corpFreq_w = \sum_{d \in docs} weightFreq_{d,w} \tag{9}$$

Next, for every document we compute the log-likelihood for each of the words and if it's more then 0.15 we add the word as an associate to this document. *docAssociates* is thus a list of documents (d) and for each document a list of associates (w). One could say that a word w that is on the list for a certain document has passed the tests and is promoted to be an associate a for the document. Note that a word can be an associate for any amount of documents. In the equations (10) and (11) *llh* stands for log-likelihood.

$$docAssociates_d = \{w \mid w \in d \text{ and } llh_{d,w} \geq 0.15\}, \text{ where} \quad (10)$$

$$llh_{d,w} = 2\left(f \ln\left(\frac{f(m+n)}{m(f+g)}\right) + g \ln\left(\frac{g(m+n)}{n(f+g)}\right)\right), \text{ where} \quad (11)$$

$f = docFreq_{d,w}$

$g = corpFreq_w$

$m =$ the total number of words in doc d without stop words

$n =$ the total number of words in the corpus without stop words

And in the final stage of the training phase we assign the associates to the descriptors from our thesaurus (*descAssociates*). Whenever an associate is assigned to a certain document it is also assigned to each of the manual descriptors of that document.

A few filters apply:

- an associate has to appear at least twice for a descriptor to be considered, see equation (19).
- the associate is weighted by the total number of manual descriptors per document, see equation (15).
- associates occurring at more than 10% of the maximum frequency of all associates in the corpus are weighted down, see equation (17).
- if the final weight of an associate is smaller than 30 it is not considered, see equation (19).

$$manDesc_d = \text{ the set of descriptors which are manually assigned to document } d \quad (12)$$

$$man_d = |manDesc_d| \quad (13)$$

This is the amount of descriptors assigned to document *d*.

$$assOccurrence_{d,s,w} = \begin{cases} 1 & \text{if } s \in manDesc_d \text{ and } w \in docAssociates_d \\ 0 & \text{else} \end{cases} \quad (14)$$

This is the occurrence of an associate *w* for a descriptor *s* in document *d*.

$$descWeight_{s,w} = \sum_{d \in docs} \frac{assOccurrence_{d,s,w}}{man_d} \quad (15)$$

This is the un-normalised weight of an associate *w* for descriptor *s*.

$$AssDescriptor_w = \sum_{d \in docs} \sum_{s \in descs} assOccurrence_{d,s,w} \quad (16)$$

This is the amount of descriptors attached to associate *w*.

$$maxNorm_w = \ln\left(\frac{\max assDescriptor}{10 \cdot assDescriptor_w} + 1\right) \quad (17)$$

This is the normalisation factor to weight down often occurring associates for associate *w*.

$$finalWeight_{s,w} = descWeight_{s,w} \cdot maxNorm_w \quad (18)$$

This is the final weight of an associate *w* for descriptor *s*.

$$descAssociates_s = \{\ w \mid assDescriptor_w \geq 2 \text{ and } finalWeight_{s,w} \geq 30\ \} \quad (19)$$

This is the set of associates for descriptor *s*.

The final result of this phase is *descAssociates*, which is the collection of the descriptors of the thesaurus, each with a list of their associates and *finalWeight*, which is a list of the weights for the associates for a descriptor.

Indexing phase

In this phase we build a list of segment-weighted word frequencies for the test document which is to be indexed. If our test document were called *t* it would be the same as $docFreq_t$ from the training phase. In the indexing phase we compare this $docFreq_t$ with each of the lists for the descriptors from *descAssociates* by means of the cosine formula. That is to say, we compare the associate vector of the test document exclusively with each of the associate vectors of the descriptors for similarity and select those descriptors for the document with the best matching vectors. For vector comparison the cosine formula is used, which is a very common measure for the similarity of two vectors and is computed for the associate vector of the test document *t* and the associate vector of any descriptor *d*, with *n* being the size of both vectors, as follows:

$$\cos(\vec{t}, \vec{d}) = \frac{\sum_{a=1}^{n} t_a d_a}{\sqrt{\sum_{a=1}^{n} t_a^2} \sqrt{\sum_{a=1}^{n} d_a^2}} \quad (20)$$

Although the JRC uses a combination of this formula and two others for the final comparison, using only the cosine formula is sufficient for testing purposes. Because the coefficients of the associate vectors are all zero or above the resulting cosine is always positive and the descriptors with the largest cosine, meaning closest to 1.0, are selected as descriptors for the document.

Also here some threshold filters apply:

- a descriptor is not assigned if it has less than ten associates on its list.
- a descriptor is not assigned if it has less than four associates in common with the document.

9.4 Segment weighting in different stages

In effect term weighting is done by artificially changing the frequency of an associate in order to get it higher or lower on the list of 'important' associates. Before, it was said that the entire process consists of two phases, but on a lower technical level there are in fact three, as a document in the training phase is actually read and processed two times. The first time to accumulate the word frequencies for the whole corpus and the second time to compute the log-likelihood for the associates in the document. Therefore it is technically possible to adjust the weights for the training documents twice in different stages of the process and to adjust the weight of the test document once. Adjusting the weights for certain segments during the accumulation of word frequencies in the training corpus has some unwanted impact on the log-likelihood ratio however. If we would attempt term weighting at that point we would see a change in b in equation (11) for any associate that somewhere appears in a weighted segment. A changed b would change the log-likelihood for that associate for all its descriptors, which is unnecessarily bulky. By the latter is meant that because we find one appearance of the associate in a weighted segment in one particular document does not mean we should change its weight for all its documents — and thereby all its descriptors. The second option in this phase, making the weighting count for the total amount of words in the corpus, would change d and thereby change the log-likelihood for any associate in any document, which is equally undesirable. So, term weighting for the overall corpus frequencies does not promise to be an improvement and indeed small experiments have shown a deterioration of the F-measure.

This still leaves us with the possibility of term weighting at two points:

1. *document frequencies when computing the log-likelihood during the training phase.*
   Adjusting the frequency of one of the associates here means a change in a in equation (11), which means that if we increase the frequency the associate will get a higher log-likelihood and is therefore considered to be more 'important' in describing the document. Decreasing the frequency will have the reverse effect. The threshold filters for the training phase mentioned in chapter 9.3 are applied *after* the weighting.
2. *document frequencies of the test document during the indexing phase.*
   Adjusting the frequency of one of the associates in the test document has a direct influence on its weight, since in this phase we do not use log-likelihood but the plain frequency of an associate as its weight. The adjustments being made have no influence on a descriptor passing a threshold filter or not, as these are based on the presence of an associate not its frequency.

9.5 The segments

There are four different types of segments taken into account during the testing.

1. *title*
   The main title of the document.
2. *chapter heading*
   This is the first sentence after a chapter number, for instance for this paragraph it would be "The segments".
3. *preamble*
   This is anything, excluding the title, that comes before the operative part of the text.
4. *appendix*
   Anything in an appendix of the document.

The choice for these four types of segments was based on the perceived theoretical benefit for the result on the one hand and the practical possibilities of identifying them with the aid of key phrases.

9.6 The results of segment weighting

So far the F-measure that has been presented has been the result of tests run by the JRC. To be able to test segment weighting it has been necessary to reproduce the procedure and set up a new test environment, for technical reasons however it has not been possible to use the exact same training corpus and test corpus as the JRC does. The training corpus that was used in the tests leading to the results below was a little over one third of the size of the full corpus and as a result of this we get a different result in terms of precision and recall, and consequently also a different F-measure. The basic result in the new test environment was an F-measure at rank 5 of 0.322, which is significantly lower than the basic result from the JRC. To be able to still compare results of the JRC and the results presented below, all the results are presented in terms of relative improvement. So far, we have seen two additions to the whole indexing process which measured at the JRC: a stop list and lemmatisation. The stop list has brought a 14.3% improvement over the results without any Lithuanian pre-processing and the lemmatisation has in turn brought a 4.8% improvement over the results with a stop list.

All the results below are compared to the basic result, i.e. the result *with* lemmatisation, *with* the use of a stop word list, but *without* segment weighting. Which means every test mentioned below is done in the new test environment and measured as a relative improvement over the result of 0.322 at rank 5, which is the result when we look at the first five descriptors found by the computer.

There have been four test rounds for the weighting which will be presented below, each with their table of results and explanation. In the first round, I have weighted only during the indexing phase, so that only the associate lists of the test documents got altered. I have done this one segment at a time. In the second round I have done the same for the training phase, so that only the associate lists of the descriptors got altered. In the third round I have weighted during both phases, but still one segment at a time and finally in the fourth round, I have combined all possible weightings at once.

9.6.1 Isolated weighting during indexing

Table 9 shows the result for weighting during the indexing phase one segment at a time. This means that the weight of only one of the four types of segments is adjusted and the rest remains normal at a weight of 1.00.

weight (indexing)				F-measure improved by %
title	chapter heading	preamble	appendix	
2.00	1.00	1.00	1.00	**0.89**
3.00	1.00	1.00	1.00	**1.05**
4.00	1.00	1.00	1.00	**0.80**
1.00	0.30	1.00	1.00	**-0.02**
1.00	0.50	1.00	1.00	**0.16**
1.00	1.50	1.00	1.00	**0.00**
1.00	2.00	1.00	1.00	**-0.13**
1.00	1.00	1.25	1.00	**0.65**
1.00	1.00	1.50	1.00	**1.31**
1.00	1.00	2.00	1.00	**1.16**
1.00	1.00	3.00	1.00	**0.40**
1.00	1.00	1.00	1.10	**0.04**
1.00	1.00	1.00	1.25	**0.12**
1.00	1.00	1.00	1.40	**0.23**
1.00	1.00	1.00	2.00	**0.05**

Table 9: Results of isolated weighting during indexing

These results show that here only two of the four segments are noticeably receptive to weighting: the title and the preamble. Although rerunning the test on a second different training corpus shows that the results for the chapter heading and appendix are consistent, they are really marginal and in a practical sense negligible.

The fact that the contents of a chapter heading actually needed to be weighted *down* was a small surprise, even though it is a fact that the actual use of chapters to indicate a different topic within the text — like it is done in this report for instance — is not that common in legislative documents. What we mostly see is one big chapter containing lots of articles, each being about a few sentences in length. The actual amount of chapter headings is therefore few, so a huge benefit was not to be expected. The fact that appendices seemed to actually contain some good information and that giving them more weight improved the results— albeit by a tiny margin — also was unexpected.

An experiment has also been done with altering the weight of the first page of a document, which was actually done by weighting the first couple of hundred words from a document. Although this showed positive results, it was in fact an imprecise way of guessing the size of the preamble and as it turned out, exactly pin-pointing the ending of the preamble by use of key phrases gave a better result than weighting the first page, no matter what amount of words where chosen as an estimate of the first page. The best result was achieved with a weight of 2.0 for the first 400 words, this gave an improvement of +0.61,

while precisely targeting the preamble in every document with a weight of 2.0 gave an improvement of +1.16.

9.6.2 Isolated weighting during training

We can try to improve the associate lists for the descriptors in the same way we have improved the associate list of a document, this time by weighting during the training phase. The results are in the table below.

weight (training)				F-measure improved by %
title	chapter heading	preamble	appendix	
1.50	1.00	1.00	1.00	**0.11**
2.00	1.00	1.00	1.00	**0.97**
3.00	1.00	1.00	1.00	**0.90**
1.00	0.50	1.00	1.00	**0.24**
1.00	0.75	1.00	1.00	**0.04**
1.00	1.00	1.10	1.00	**0.20**
1.00	1.00	1.20	1.00	**0.77**
1.00	1.00	1.50	1.00	**0.60**
1.00	1.00	2.00	1.00	**-0.55**
1.00	1.00	1.00	1.50	**0.78**
1.00	1.00	1.00	2.00	**0.80**
1.00	1.00	1.00	2.50	**0.82**
1.00	1.00	1.00	3.00	**0.10**

Table 10: Results of isolated weighting during training

For the title, chapter heading, and preamble the results are in the same order of magnitude as the results for isolated weighting during indexing. An associate list that is built for a descriptor will usually be based on much more data than an associate list for a single document, as a descriptor is usually assigned to more than one document. The more data we have, the better our statistics will be, and the better also our associate lists will be. This fact might explain the results for the title and preamble to be slightly lower — seeing as there is a smaller room for improvement —, but given the relative small difference this could also be a chance aberration.

The results of giving more weight to appendices are surprisingly higher when compared to those for indexing. The amount of documents with an appendix is somewhat higher in our test corpus than in our training corpus, 35% of the documents in the test corpus has an appendix to it against 28% in the training corpus. To make a definitive conclusion whether or not there is some causality between this difference and the difference in test results would require more research on a different corpus, which I have refrained from doing due to time restrictions.

9.6.3 Isolated weighting in both phases

In these tests weights were adjusted during both the training *and* the indexing phase, still only one segment at the time. The first part of the weight in the table is the one used during training, the second part the one used during indexing.

weight (training / indexing)				F-measure improved by %
title	chapter heading	preamble	appendix	
2.00 / 2.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	**1.05**
2.00 / 3.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	**1.19**
2.00 / 4.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	**0.93**
3.00 / 2.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	**1.38**
3.00 / 3.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	**0.94**
1.00 / 1.00	0.50 / 0.50	1.00 / 1.00	1.00 / 1.00	**0.24**
1.00 / 1.00	0.50 / 0.75	1.00 / 1.00	1.00 / 1.00	**0.17**
1.00 / 1.00	1.00 / 1.00	1.10 / 2.00	1.00 / 1.00	**1.15**
1.00 / 1.00	1.00 / 1.00	1.20 / 2.00	1.00 / 1.00	**1.10**
1.00 / 1.00	1.00 / 1.00	2.00 / 2.00	1.00 / 1.00	**0.63**
1.00 / 1.00	1.00 / 1.00	2.00 / 3.00	1.00 / 1.00	**0.04**
1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	2.00 / 2.00	**0.58**
1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	2.50 / 2.00	**0.41**

Table 11: Results of isolated weighting in both phases

One of the conclusions we can draw from the two previous tables 9 and 10 is that the effect of changing weights is largely independent between the two phases. The mutual independence is not complete, but still clearly shows from the fact that we see a better result from only improving the quality of our associate lists for descriptors in the training phase or only improving the quality of the associate list for a document in the indexing phase. Quality here means the accuracy of an associate list in describing either a descriptor or a document. This independence is not entirely obvious because the outcome of the automatic indexing is the result of a comparison for similarity of the descriptor associate lists with the document associate list. In other words, if we improve the quality of the descriptor associate list, but still compare it to a document associate list of lesser quality we still might not get a better match. That nonetheless we *do* see some document better matching the right descriptors when we increase the quality of either associate lists, is a sign that the associate lists gathered in either phase have a good quality of their own. This independence further shows itself in the *extra* improvement we get when we weight in *both* phases. For instance the best result for the title in Table 11 is 1.38, while it was 0.97 and 1.05 for training and indexing respectively. That there still is some interdependence between the two types of associate lists can be seen in the same results for the title, as the weights giving us the best combined result are not the very same as the weights giving us the individual best results. Indexing is best at 3.00 and training at 2.00, whereas the combined weight that gives us the best result is the other way around: 2.00 for indexing and 3.00 for training.

The only thing that now remains to be seen is if weighting for multiple segments at the same time again provides a surplus improvement over the results we have seen so far.

9.6.4 Combined weighting

weight (training / indexing)				F-measure improved by %
title	chapter heading	preamble	appendix	
1.00 / 2.00	1.00 / 0.50	1.00 / 1.50	1.00 / 1.40	**1.53**
1.00 / 3.00	1.00 / 0.50	1.00 / 1.50	1.00 / 1.40	**1.20**
2.00 / 1.00	0.50 / 1.00	1.20 / 1.00	2.50 / 1.00	**0.15**
2.00 / 1.00	0.50 / 1.00	1.00 / 1.00	1.00 / 1.00	**0.69**
1.00 / 1.00	1.00 / 1.00	1.20 / 1.00	2.50 / 1.00	**0.48**
2.00 / 2.00	1.00 / 0.50	1.00 / 1.50	1.00 / 1.40	**2.01**
1.00 / 2.00	1.00 / 0.50	1.20 / 1.50	1.00 / 1.40	**1.29**
1.00 / 2.00	1.00 / 0.50	1.00 / 1.50	2.50 / 1.40	**1.64**

Table 12: Results of combined weighting

If we take the best weight of every segments from isolated weighting during the indexing phase and apply these all at the same time, we see a small extra improvement over the isolated weightings (1.20). Interdependence between these segments is very likely, as they are very likely to share some associaties that they give more weight. With combined weighting these associates get even more weight and that might be overdoing it a little. This means we could get some more improvement by tweaking some more with the weights, for instance a weight of 2.00 for the title already gives us a further improvement to 1.53.

Combined weighting during the training phase with the best weights from isolated weighting gives a completely different result however. So far, weighting during training and indexing has produced similar results, but this equality stops here. It seems interdependency between the segments here is really strong, as the combinations of weighting — these are rows three, four and five in Table 12 — all give worse results than isolated weighting for just one of the segments. A mix of isolated weighting for the training phase with combined weighting for the indexing phase does however give extra improvement.

It seems any combination of weights gives more result than the highest of the individual results in that combination, except for combining the segments during training. A possible explanation for this, is that where the indexing segments had *some* overlap in the associates that get more weight, the training segments have even more overlap, up to the point where overweighting these associates they share almost nullifies the results of isolated weighting.

9.6.5 Results in recall and precision and at other ranks

In the tables above results are constantly depicted in the F-measure at rank 5, but what happens under the hood of the F-measure, i.e. what are the effects of weighting on recall and precision, and what happens at other ranks? Recall and precision behave consistent across all weights, phases and ranks, both growing at the same rate as the F-measure does. This means that weighting has the same positive effect on recall as it has on precision. This in contrast to the stemming which only increased recall at a small loss of precision.

When we look at the results of weighting at different ranks we see a downward slope in the results when we increase the rank of measurement. Meaning results are best at ranks 1 and 2 and decrease to a nil improvement at a rank of 10. What this means is that on average our top 10 of retrieved descriptors does not change at all. The improvement we see

at for instance a rank of 5 is most likely due to a reshuffle of the top 10 descriptors to become a better top 5. For example, a good descriptor — one that was also assigned manually — was at place 6 before weighting, so when looking at rank 5 it got cut off. After the weighting it has switched places with a bad descriptor — a descriptor *not* assigned manually — and is now at fifth place making the bad descriptor stand at place 6. If in this new situation we look at rank 5, we will see a better results, but if we look at rank 10 nothing has changed, because both the good and the bad descriptor are still in the top 10. In other words, the segment weighting based on predefined structure favours associates that were already considered good associates, merely altering their order of ranking.

The complete set of test results can be found on the cd-rom that accompanies the printed version of this thesis.

10 Automatic segmentation

With semi-manual segmentation I have explored the possibilities of using the extra information that lies in the predefined structure of the texts. In other words it was based on the assumption that one segment of a document contains more information on its topic than another segment and that the alteration of these more and less important segments is parallel to the alterations in the predefined official structure of the document. Indeed the results presented in chapter 9 show that some of these predefined segments contain more information than others. However the largest part of the text, namely the operative part, is still treated as a whole, while it does not necessarily have to be one long homogeneous stretch of a text in which every chapter or article contains as much information as the other. It is possible to divide the operative part into chapters and articles with the use of key phrases, but as was explained in chapter 9.6.1 the use of chapters is not abundant and the articles are too many and too small in size to serve as segments for topic differentiation. So if our knowledge of the predefined structure falls short, perhaps it is possible to reach for the information that lies in the text itself, i.e. the statistics of the occurrence of words in it, for this we could use automatic segmentation.

Automatic segmentation is aimed at fully automatically dividing a text into segments, so that each of these segments is homogeneous and represents a topic that stands out from its surrounding segments. That is to say that automatic segmentation does not make use of something like key phrases or other such provided information on a predefined structure of the document. The first attempt at this was the Text Tiling algorithm developed by Hearst and Plaunt [12] in 1993. Although they showed the possibilities the results weren't good enough to be implemented and used in actual working systems. Since then, however, different and better algorithms have been put forward and although all of these are still under development, they might give a good indication of the benefits of automatic discourse segmentation.

The specific algorithm discussed here is the one developed by Fragkou, Petridis and Kehagias [7]. The purpose of their algorithm is to make a linear segmentation of a text, as opposed to a hierarchical segmentation. This means we do not divide sections into subsections, for instance a preamble which is subdivided into chapters and a chapter in turn subdivided into clauses, but we keep it all on one level. The choice for linear segmentation has to do with the fact that at this point there are better algorithms available for this type of segmentation, but more importantly because the kind of texts that are within the scope of blind segmentation will most probably not have a hierarchical lay-out, since this type of lay-out is usually explicitly present in the text by means of some type of heading in which case we would be able to use the much more efficient smart segmentation. The result of a linear segmentation is a list with breakpoints for the text, these are the points where one segment ends and another one begins.

The optimal segmentation of the whole text is calculated by comparing *possible* segments on two factors: 1) similarity of words within each segment, and 2) prior statistical information about the average length of a segment. To improve our outcome we can shift the weight we put on each of these two factors, where the most weight will usually be on the first factor[1].

[1] According to Fragkou et al. the best results have been achieved with a 60% to 92% weight for factor 1.

10.1 First factor: word similarity

The factor of word similarity is based on the general belief that parts of a text that have a similar vocabulary will most likely concern the same topic. So if a group of successive sentences share a vocabulary and the surrounding text does not share that vocabulary we can say that this group of sentences forms a segment within the text. Similarity between sentences across the whole text is represented by a sentence similarity matrix. The sentence similarity matrix is constructed as follows: First as an intermediate step we build a matrix F, in which all sentences of the text are plotted against all words in the vocabulary of the text.

F is defined as:

T = number of sentences in text
L = number of words in vocabulary
for $t = 1,2,...,T$ and $l = 1,2,...,L$ we set

$$F_{t,l} = \begin{cases} 1 & if\, the\, l^{th}\, word\, appears\, in\, the\, t^{th}\, sentence; \\ 0 & else. \end{cases} \qquad (21)$$

In plain English: for each sentence we check all the words of the vocabulary, if the word is in the sentence we jot down a 1, if it is not we jot down a 0, this way we end up with a long list of 1's and 0's for each sentence. To illustrate this, let us take the following four sentences:
1. *the apple is red*
2. *a red apple is sweet*
3. *pears are mostly green*
4. *pears are fruity*

They would lead to the following matrix F:

	sentence 1	**sentence 2**	**sentence 3**	**sentence 4**
the	1	0	0	0
apple	1	1	0	0
is	1	1	0	0
red	1	1	0	0
a	0	1	0	0
sweet	0	1	0	0
pears	0	0	1	1
are	0	0	1	1
mostly	0	0	1	0
green	0	0	1	0
fruity	0	0	0	1

Table 13: Example F matrix

With matrix F in our toolbox, we proceed to constructing the sentence similarity matrix D, which is a symmetrical matrix of all sentences plotted out against themselves, i.e. T x T.

D is defined as:

for s,t = 1,2,...,T we set

$$D_{s,t} = \begin{cases} 1 & if \quad \sum_{l=1}^{L} F_{s,l} F_{t,l} > 0; \\ 0 & if \quad \sum_{l=1}^{L} F_{s,l} F_{t,l} = 0. \end{cases} \tag{22}$$

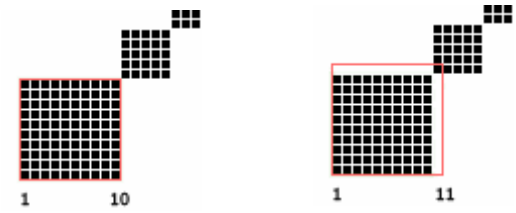
In plain English: Two sentences are marked with a 1 if they share at least one word from the vocabulary. We can see in our example that sentences 1 and 2 share words — outlined in Table 13 — so we put a 1 at the point where they cross in the matrix D and the same goes for sentences 3 and 4. Of course every sentence shares all words with itself so at the diagonal we get all ones. The D matrix from our example thus becomes:

4	0	0	1	1
3	0	0	1	1
2	1	1	0	0
1	1	1	0	0
	1	**2**	**3**	**4**

Table 14: Example D matrix

In this example it is still pretty clear to see that there is a cluster of 1's near sentences 1 and 2 and a similar cluster near sentences 3 and 4. If we have a normal text however with a lot of sentences a D matrix can become huge and hard to interpret for the human eye. In that case we could draw a so-called *dot plot* of the matrix. In a dot plot we represent the matrix by drawing a little black square — a 'dot' — where we find a 1 in the matrix and leave the space white if we find a 0. This way we can visualise the similarity of sentences across the text in a way that is much easier to interpret with the naked eye, for an example of a dot plot for a text with eighteen sentences see Figure 5. The picture we see in Figure 5 is quite an ideal one where there are large black clusters along the diagonal axis that indicate regions with a high mutual similarity. We can see that these clusters of dots are most likely segments within the text, so all we need is a way to automate the process of finding the breakpoints. A breakpoint is a point in the text where one segment ends and another one begins. Automatically indicating these breakpoints can be done by defining a cost function for the segmentation, thereby describing it as an optimisation problem and then have the computer solve it. The exact formula for this cost function can be found in Appendix I. It suffices to say here that the process of finding the breakpoints of the segments is the same as finding the local maxima of the density. This goes as follows: we start off by looking at the text through a window, starting at a window of 1 sentence and increasing the window with one sentence each step. In Figure 5 is illustrated what happens when we reach a breakpoint, the square outline is our current window. In Figure 5.a we look at a window of the first 10 sentences and the word similarity density is still at its maximum. Our window is 10x10=100 units large and there is the same amount of dots in it, so the density is at 100%. Next we increase our window with one sentence to 11 (Figure 5.b) and the density for the whole window drops, as we now have a window of 121 units with 101 dots in it making a

density of about 83%. Now we know we have reached a local maximum and we can set a segment breakpoint at sentence number 10. We continue to do this until the end of the text, which gives us a list of all the breakpoints.



(a) (b)

Figure 5: finding breakpoints

10.2 Second factor: segment length

The matrix used for Figure 5 is of course oversimplified. In reality breakpoints are not that clear-cut so we could use some extra information to base our decision on. It could happen for instance that when we increase the window from 2 to 3 sentences we find that the density drops, the computer would then mark that as a breakpoint but some common sense tells us that this is most likely a coincidence, because 2 sentences is way too short for a segment. So, what we want to do is to transfer this 'common sense' in some way to the software. Fragkou e.a. solve this by taking the average length of a segment and also the deviation from this length into account. The average length of a segment would have to be estimated beforehand based on corpus research and can be measured in terms of words or sentences. With this information we can tell the computer to assign a breakpoint only if this results in a segment that is not far smaller than the average.

10.3 Possibilities of automatic segmentation

When this technique is successfully applied we end up with a document that is divided into segments that each stand out from their *surrounding* segments, and next we could give each of these segments a different weight. However, at this point we still don't know how to weight each of the segments. This problem could be solved by using an iterative approach to the weighting of these segments. Iterative term re-weighting is often applied in improving queries to a database by interacting with the user. For instance, a user has sent a query to a database and is looking at the results, this user is then asked to rate the results. If the results would contain ten documents, the user is asked to tell which of the documents conform to what her or she was looking for and based on this information the query is modified automatically and the new query gets sent again to the database to retrieve more of the good type of documents.

The same thing can also be done by re-evaluating the first results not by feedback of the user, but by looking at the results of the individual segments. This type of term re-weighting based on a computer-driven evaluation of the first iteration is also not uncommon in statistical research. Let us say we have a document and it is divided into four segments by means of automatic segmentation. We would then first try to build an associate list for the document with all the four segments at the same weight, i.e. in the proportion of 1:1:1:1. Next we would build an associate list for each of the segments, and then compare each of these lists separately to the list for the whole document, by means of the cosine method. In

other words, we would treat each of the four segments as if it was a small document on its own and compare them to the document as a whole. Based on this comparison we would re-weight the segments and perform a second iteration of building the associate list and afterwards proceed with the indexing. For example, if the second and third segment have the highest agreement with the associate list of the whole document, we would change the proportion of the segments to 1:2:2:1 and then recalculate all the frequencies in the same way we do for a title or appendix that has gotten a different weight. In this example the weight for segment two and three is doubled, but the weight could instead be related to the outcome of the cosine comparison for each of the segments. The higher the cosine, the higher the weight in the second iteration. In theory two iterations would be sufficient, because with more iterations we would see a snowball effect in which the segments with a higher weight would keep getting more agreement with the associate list of the whole document, which leads to a higher weight, which leads to a higher agreement, et cetera.

Although admittedly the results of the semi-automatic segmentation are not encouraging for the potential of segmenting as a whole, the technique of automatic segmenting as discussed above does produce an entirely different kind of segmenting. With semi-automatic segmenting, minor portions of the text were targeted based on the idea that their structural division would correlate to a topical division, in other words that their place in the structure would say something about their place in the content. With automatic segmentation however, an attempt is made to create a segmentation that is *based* on the content not on the structure. In other words we create a new structural division that directly follows from the topical division. And we would do this for the operative part of the text and the appendix, which form the bulk of the text containing the most data, but each of the two has so far been treated as a whole. With automatic segmentation we would be addressing more data with a different approach, which could lead to some different results than semi-automatic segmentation.

11 Conclusions and recommendations

The central question in this thesis was if and how pre-processing can help to improve statistical analysis tasks for Lithuanian legislative text. In my research I have focussed specifically on the automatic indexing of Lithuanian documents as it is done by the JRC within the context of the Eurovoc project of the European Union. With the growth of the number of members and the range of duties of the EU, the bureaucracy in Brussels is expanding rapidly and with it, the need for fluent communication across all languages, especially concerning legislative document as these are the foundations of our national and international policies.

Each of the judicial languages can, according to the literature, be considered a language within a language. This fact makes translating texts within a context of law even more difficult than normal translation. Different uses of words and concepts, differences in notation or even in the approach of the administration of law as a whole, all attribute to the complexity of translation. Large and easily accessible reference corpora form an essential tool to translators and cross-lingual search-engines like those provided in conjunction with the Eurovoc thesaurus can play an important role in providing exactly that. All the more now that 10 new languages have joined the Babel-like confusion, and that it can be concluded that within the EU offices and courts a new European judicial language is developing.

My research in improving the results for automatic indexing for Eurovoc can be divided into two major areas: *morphology* and *document structure*. In some places I have found it safe to extend my conclusions beyond the scope of the Eurovoc project and/or the Lithuanian language and in others they are strictly confined to their specific context.

11.1 Morphology

The basic results for automatic indexing without pre-processing are lower for the Lithuanian language than for most other languages. The hypothesis that this is related to the rich morphology of the language is backed up by literature on this topic as well as the positive results that are gained from pre-processing with an inflectional stemmer.

Concerning the stemmer I have found that a rudimentary stemmer for nouns, adjectives and pronouns that does not result in grammatically correct stems already provides a significant improvement for the results of automatic indexing and that at the same time more grammatical correct stemming or extending the stemmer to include verbs or derivational suffixes is not very likely to produce much better results. The improvements of the stemming were consistent at various ranks and were entirely due to an increase in recall, while precision dropped very slightly. Combined with the fact that enhancing the stemmer in any of the before mentioned ways will require a lot of effort for a small prospected gain, I would be inclined to say that from a practical point of view it would not be advisable to create a more refined stemmer. In its current set-up the stemmer could enhance statistical analysis tasks for Lithuanian that operate on large corpora and which do not require morphologically correct stemming per se. In fact, the same type of stemmer could be built based on a list of suffixes for other languages with a rich inflectional morphology and is very likely to produce similar positive results.

11.2 Document structure

The legislative documents in the corpus that was used in this research and that of the JRC adhere to a very strict predefined structural lay-out. The initial hypothesis behind segmenting the documents based on this structure, was that a topical division would run parallel with this structural division and that the building of associate lists could be improved by isolating these structural segments in the text and giving them separate weights. From my test results with this semi-automatic segmentation it can be concluded that in some cases the results indeed improve when segments are treated different from each other during the building of associate lists, and thus that there is indeed a correlation between the structural divisions and topical divisions in the sense that some specific types of segments will contain more words that give away the topic than other segments. These 'more important' segments are the title, the preamble and the appendices. Tests with targeting the chapter headers did not lead to satisfactory results, but this was not a surprise given the fact that chapter headers are scarcely used in the type of legislative documents contained in our testing corpora.

There were no significant differences in applying the weighting when building associate lists in the training phase for the descriptors and in the indexing phase for a document. Except for the appendices, which seemed to benefit more from weighting in the training phase. A search for a satisfactory explanation for the latter remained inconclusive.

It appeared very much possible to improve either the associate lists generated in the training phase or the ones generated during indexing by weighting segments exclusively, and still improve the overall result. Therefore it can be concluded that we can speak of associate lists as having a quality that can be improved to give better results during comparison with the cosine method, and that this quality is to a certain extent independent of the quality of the other associate list in this comparison. When isolated weighting is used in both phases this quality independence also leads to an extra improvement over the results of isolated weighting in only one phase.

Combined weighting for more than one segment at a time proved to be benificial too, except for combining segments during training, which actually worsened results. An explanation for this can be sought in an overlap of benefiting associates among the segments.

The improvements after weighting in any rank, any phase and on any segment were the same for precision and recall. Results were best at lower ranks and dropped as we measure at an increasing rank, to become zero at a rank of ten. From this, we can conclude that segment weighting based on predefined structure does not result in a different top ten of descriptors, but does lead to a better ordering of this top ten. Whether this is useful or not depends on the context in which the automatic indexing is applied.

Finally, the technique of automatic segmentation I have discussed could lead to some interesting results, because it is an attempt to differentiate segments in the largest part of the data in our documents, namely the operative part and the appendices, by looking at the words they contain. Although this technique is still not perfected it would be interesting to see if re-weighting based on a topical division could lead to different results than re-weighting based a predefined structural division.

Bibliography

[1] B. Brodda, F. Karlsson, “An experiment with automatic morphological analysis of Finish”, 1980, in Papers from the Institute of Linguistics University of Stockholm, vol. 40

[2] E. Bucher, “The position of the civil law of Turkey in the western civilisation”, in international conference ‘Atatürk and modern Turkey’, Ankara University, 1999

[3] F.H. van der Burg, Europees gemeenschapsrecht in de Nederlandse rechtsorde, Kluwer, Deventer, 2003

[4] I. Burr, G. Gréciano, “Europa: Sprache und Recht, La construction européenne: aspects linguistiques et juridiques”, Nomon Verlagsgesellschaft, Baden-Baden, 2003

[5] Cologne European Counsil, “Annex IV, Annexes to the precidency conclusions”, Cologne, 3 and 4 june 1999

[6] N.A. Florijn, N.A., “Rechtstaal, lexicon en vertaling”, in “Recht en vertalen II” edited by G.R. de Groot, Deventer 1993

[7] P. Fragkou, V. Petridis, Ath. Kehagias, “A dynamic programming algorithm for linear text segmentation”, 2002, in “Journal of Intelligent Information Systems”, vol. 23, no. 2

[8] “De beschermde status van 20 officiële talen”, http://www.grondweteuropa.nl/9326000/1f/j9vvgjnazrhmix9/vgonesxsmbu2, March 17 2005

[9] G. R. de Groot, “Een nieuw tweetalig woordenboek” in Recht en vertalen II, Kluwer, Deventer, 1993

[10] G. R. de Groot, “Het vertalen van juridische informatie. Rechtsvergelijking in de bestudering van het publieksrecht: een stillistische benadering”, Kluwer, Deventer, 1996

[11] J.C. Hage, R. Wolleswinkel, “Recht, vaardig en zeker”, Boom juridische uitgevers, Den Haag, 2003

[12] M. A. Hearst, C. Plaunt, “Subtopic structuring for full-length document access”, 1993, in Proceedings of SIGIR '93, p. 59

[13] M. C. Hoadly, “The role of law in contemporary Indonesia”, Centre for East and South-East Asian Studies Lund University, Sweden, 2004

[14] http://www.jrc.cec.eu.int, 2004

[15] A. Klimas, “The importance of Lithuanian for Indo-European linguistics”, in Lituanis, vol. 15, no.3, 1969

[16] W. Kraaij, R. Pohlmann, “Viewing stemming as recall enhancement”, 1996, in Proceedings of the 19th conference on research and development in Information Retrieval, p. 40

[17] A.J.C de Moor - van Vugt, “Europees bestuursrecht”, Tjeenk Willink, Deventer, 1998

[18] W. Pintens, “Rechtsvergelijking en taal” in Ars Aequi vol. 43 (1994), no. 5, pag. 290-296

[19] M. Popovic and P. Willett, “The effectiveness of stemming for natural language access to Slovene textual data”, 1992, in Journal of the American Society for Information Science, vol. 43, no. 5, p. 384

[20] M.F. Porter, “An algorithm for suffix stripping”, 1980, in Program, vol. 14, no. 3, p. 130

[21] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, “Okapi at TREC-3”, Proceedings of Text Retrieval Conference TREC-3, U.S. National Institute of Standards and Technology, Gaithersburg, USA, pp. 109-126

[22] J. Smits, “Europees privaatrecht in wording: Naar een Ius Commune Europaeum als gemengd rechtstelsel”, Intersentia uitgevers, Antwerpen, 1999

[23] J. Smits, “The contribution of mixed legal systems to European private law”, Intersentia uitgevers, Antwerpen, 2001

[24] D. Soergel, “Multilingual thesauri in cross-language text and speech retrieval”, In AAAI Symposium on Cross-Language Text and Speech Retrieval. American Association for Artificial Intelligence, March 1997

[25] R. Steinberger, B. Pouliquen, “Cross-lingual indexing”, Final report for the IPCS exploratory research project, Ispra, March 2003

[26] M. Termorshuizen-Arts, “Juridische semantiek”, Wolf legal publishers, Nijmegen, 2003

[27] V. Zinkevicius, “Lemuoklis: morfologinei analizei”, 2000, in Darbai ir Dienos, no. 24, p. 245

Appendix I – Formulae

The Okapi formula

The Okapi formula, from Robertson [21], uses the term frequencies and the length of descriptors (i.e.: the number of associated lemmas).

$$Okapi_{t,d} = \sum_{l \in t \cap d} \log\left(\frac{N - DF_l}{DF_l}\right) \frac{TF_{l,d}}{TF_{l,d} + \frac{|d|}{M}}$$

With:

d	The descriptor
t	The new text
l	A lemma (keyword)
N	The number of Eurovoc descriptors (to be precise: this is the number of "trained" Eurovoc descriptors)
|d|	The size of the descriptor (number of associates, which is 400 maximum)
M	The average size of descriptors (average number of associates per "trained" descriptor)

The length of the text is not taken into account, but, in our case, this does not have any impact because we use this formula to order the results for the same text.

The automatic indexing cost function with only word similarity

$$J(\mathbf{t}; r) = \sum_{k=1}^{K} \left(\frac{\sum_{t=t_{k-1}+1}^{t_k} \sum_{t=t_{k-1}+1}^{t_k} D_{s,t}}{(t_k - t_{k-1})^r} \right)$$

t	The vector containing segment boundaries
r	When r=2 the term between parenthesis is the density of the segment, when r≠2 this becomes a 'generalised density'
K	The (variable) length of vector **t**
$D_{s,t}$	The sentence similarity matrix

The complete automatic indexing cost function including segment length

$$J(\mathbf{t}; \mu, \sigma, r, \gamma) = \sum_{k=1}^{K} \left(\gamma \cdot \frac{(t_k - t_{k-1} - \mu)^2}{2 \cdot \sigma^2} - (1 - \gamma) \cdot \frac{\sum_{t=t_{k-1}+1}^{t_k} \sum_{t=t_{k-1}+1}^{t_k} D_{s,t}}{(t_k - t_{k-1})^r} \right)$$

μ	The average segment length
σ	The standard deviation of the segment length
γ	A parameter to weight the relative importance of the word similarity and segment length