UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

DIPLOMA THESIS num. 1683

Collocation extraction measures for text mining applications

Saša Petrović

Zagreb, September 2007

This diploma thesis was written at Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, and at Institut de recherche en informatique et systèmes aléatoires (IRISA), Université de Rennes 1, Rennes, France, during INRIA internship from April 12 to June 12, 2007.

Contents

List of Figures

List of Tables

List of Examples

Acknowledgments

The work presented in this thesis would never be completed without the help of many people. First of all, I would like to thank my advisor, prof. Bojana Dalbelo Bašić for her patient guidance and helpful comments, and to Jan Šnajder for many useful advices, before and during the writing of this thesis.

Part of the work on integrating the text mining module in Orange was done at the AI Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Slovenia. There, prof. Blaž Zupan, prof. Janez Demšar, Gregor Leban, Frane Šarić, and Mladen Kolar all helped to get the first version of the text mining module up and running.

Completing the work on correspondence analysis would not be possible without prof. Annie Morin, who supervised me during my internship at IRISA, Rennes, France, and to whom I am very thankful for that and for everything else she has done for me.

Finally, I would like to thank my girlfriend for the support and understanding during the writing of this thesis, and to my mother whose continuing love and support helped me become the person I am today.

Part I

Collocation Extraction

CHAPTER 1

Introduction

In the beginning was the Word,
and the Word was with God, and
the Word was God

Bible

Natural language processing (NLP) is a scientific discipline combining the fields of artificial intelligence and linguistics. It studies the problems of automated generation and understanding of natural human languages and uses linguistic knowledge, namely grammars, to solve these problems. *Statistical NLP* is a specific approach to natural language processing, which uses stochastic, probabilistic, and statistical methods to resolve some of the difficulties traditional NLP suffers from. For example, longer sentences are highly ambiguous when processed with realistic grammars, yielding thousands or millions of possible analyses. The disambiguation in statistical NLP is carried out with the use of machine learning algorithms and large corpora. *Text mining* is a subfield of *data mining*, which is, as NLP, a subfield of artificial intelligence. Text mining is an interdisciplinary field, combining fields like *information retrieval, machine learning, statistics,* and *computational linguistics.* Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling. *Collocation extraction* is one of many tasks in statistical NLP, and it involves finding interesting word combinations, collocations, in large corpora. Collocation extraction will be the topic of the first part of this thesis. The second part of the thesis will describe the application of collocations on one text mining task—visualization of a corpora. The goal of this visualization will be to find clusters of documents that talk about similar topics.

1.1 What are collocations and what have they done for me lately?

A collocation is "an expression consisting of two or more words that correspond to some conventional way of saying things" [32]. Even though the previous definition gives some insight into what a collocation is, it fails to give a precise and formal definition of the term collocation that could be used in real applications. Closely related to the term collocation is the term word n-gram, which denotes any sequence of n words. A word n-gram consisting of two words is called a digram, word n-gram consisting of three words is called a trigram and a word n-gram consisting of four words is called a tetragram. In the first part of the thesis, for simplicity reasons, instead of writing "word n-gram", the term "n-gram" will be used.

Over the years, many authors tried to give a definition of a collocation, but even today there does not exist a widely accepted one. Various definitions range from identifying collocations with idioms, to saying that a collocation is just a set of words occuring together more often than by chance. However, there are three criteria which most collocations satisfy [32]:

- **Non-compositionality** means that the meaning of the whole collocation is more than a sum of meanings of the words forming it.
- **Non-substitutability** means that we cannot substitute a word in a collocation with another word having similar or even same meaning.
- **Non-modifiability** means that we cannot freely modify the collocation with additional lexical material or put the collocation through some grammatical transformations. This criteria is especially true for idioms.

The definition of collocation adopted here lies somewhere in between. By a notion of collocation four different types (subclasses) of collocations will be considered. The first one coincides with the definition of an *open compound* (*compound noun*) in [46]. An open compound is defined as an uninterrupted sequence of words that generally function as a single constituent in a sentence (e.g., *stock market, foreign exchange*, etc.). The second and third types of collocations covered here are *proper nouns* (*proper names*) and *terminological expressions*. The latter usually refers to concepts and objects in technical domains (e.g., *monolythic integrated circuit*). The fourth type of collocation is somewhat less idiomatic and more compositional than an open compound and it involved sequences of words often occuring together interrupted by a preposition or a conjunction, and describing similar concepts (e.g., *sport and recreation, guns and ammunition*, etc.). We should note here that all these types of collocations are uninterrupted and short-span, unlike long-span collocations used in [55].

There are many possible applications of collocations [32]: finding multiple word combinations in text for indexing purposes in information retrieval, automatic language generation, word sense disambiguation in multilingual lexicography, improving text categorisation systems, etc. These applications will be discussed further in the next section.

The motivation behind the whole process of extracting collocations described here was improvement of the document indexing system CADIS [30]. This system was initially developed for indexing documents in Croatian, so that is why, in this work, more weight is given to extracting collocations from Croatian corpora. The reason for adopting the definition of a collocation mentioned above now becomes apparent. All four mentioned types of collocations are very useful for indexing purposes—the first three types are known to bare useful information about content of a document, while the fourth type adopted here was found very useful for indexing performed by human experts. The focus of this work is to filter out non-collocations that could not otherwise be filtered out by POS tags and frequency alone. It is hoped that the extracted collocations will help improve the indexing system [30] by serving as a complement to the traditional bag-of-words representation of a document. For example, if the word *foreign* appears 10 times in some document, one can tell very little about the topic (content) of the document. But, if the collocation *foreign exchange* appears 10 times in some document, the document is probably about economics or money in general, while if the collocation *foreign student* appears frequently, the document is probably about education.

1.2 Related Work

There are a lot of papers that deal with the problem of collocation extraction, but the lack of a widely accepted definition of a collocation leads to a great diversity in used measures and evaluation tehniques, depending on the purpose of collocation extraction. Smadja and McKeown [46] use collocation extraction for the purpose of language generation, so they seek to capture longer collocations and especially idioms in order to improve their system. They use a lot of statistical data (word frequencies, deviation, distances, strength, etc.) to accomplish the task. On the other hand, Goldman and Wehrli [20] use their system FipsCo for terminology extraction, so they rely on a very powerful syntactic parser. Unlike both of them, Wu and Chang [58] set out to extract collocations from a bilingual aligned corpus, and for this they use a number of preprocessing steps in combination with the log-likelihood ratio and a word alignment algorithm, while Vechtomova [55] uses long-span collocations for query expansion in information retrieval.

In order to compare AMs, a framework for evaluating them is needed. Unfortunately, there doesn't exist a method for evaluating AMs on which a majority of authors agree so there are a number of different approaches used by different authors. For example, Smadja [46] employs the skills of a professional lexicographer to manually tag n-grams as either collocations or non-collocations, Thanopoulos et al. [52] and Pearce [39] use WordNet [19] as a gold standard, while Evert and Krenn [17] use a small random sample of the entire set of candidates for comparison. Somewhere in between lies the approach taken by da Silva and Lopes [45]. They have manually inspected several hundred randomly selected n-grams from the set returned by each of the tested measures, tagged them as collocations or non-collocations and computed precision based on that. Each of these methods has its advantages and its problems—Smadja's approach gives a very accurate value for precision and recall but on the other hand takes very long, Thanopoulos' method is faster but, as he states "WordNet is both impure and incomplete regarding non-compositional collocations", while Evert's method is the fastest one and good for ranking AMs, but one can only estimate true recall and precision for an AM. The confidence intervals for the estimate will then depend on the size of the random sample. With the method used by da Silva and Lopes it is impossible to compute recall so they use the total number of multi-word units extracted by each measure as an indirect measure of it. Method of evaluation adopted here is similar to Evert's and will be thouroughly described in section 4.

The work undertaken in the first part of this thesis is actually an extension of work done by Petrović et al. [41]. In [41], a basic framework for the experiments was established and experiments were run on the Narodne Novine corpus, for digrams and trigrams. Here, the experiments are extended to three more corpora, and also to tetragrams. In addition, the work done here on extending the association measures is completely new.

First part of the thesis is organized as follows:

Chapter 2 describes a formal approach to corpus preprocessing.

Chapter 3 gives an introduction to the used association measures, their possible extensions for trigrams and tetragrams, and also proposes some heuristic ways of extending them.

Chapter 4 describes the datasets and the approach to evaluation in more detail.

Chapter 5 gives the results and discusses them, while

Chapter 6 outlines the possible future work and concludes the first part.

The second part has the following structure:

Chapter 7 explains what are letter n-grams and where they are used.

Chapter 8 gives the mathematics behind correspondence analysis, the tool used to visualize the corpora.

Chapter 9 describes how the different text preprocessing methods are implemented in a data mining software called Orange, and also how to use them.

Chapter 10 compares how the different text features perform on the task of visualizing the corpora in order to find if some of the features are better than others for this.

Chapter 11 concludes the second part.

CHAPTER 2

Corpus Preprocessing

Words are more treacherous and powerful than we think

Jean Paul Sartre

Collocations are extracted according to their ranking with respect to an association measure. These measures are based on raw frequencies of words and sequences of words (n-grams) in corpus, which are obtained by preprocessing the corpus. In this context, preprocessing the corpus means tokenization, lemmatization, and POS tagging of the words in the corpus, and counting how many times each word and n-gram appears in the corpus.

In this chapter, preprocessing of the corpus will be formalised, which is not usually done in literature. The reason for including this formalisation, taken from [41], is that it enables later definition of extensions for association measures.

2.1 Obtaining Word n-grams

Definition 2.1. *Let* W *be a set of words and* P *be a set of punctuation symbols, and* $\mathrm{W} \cap \mathrm{P} = \emptyset$. *The corpus* C *is represented as a sequence of tokens, i.e., words and punctuation symbols, of finite length* k*:*

$$\mathrm{C} = (t_1, t_2, \ldots, t_k) \in (\mathrm{W} \cup \mathrm{P})^k. \tag{2.1}$$

Let $\mathrm{W}^+ = \bigcup_{n=1}^{\infty} \mathrm{W}^n$ *be the set of all word sequences. An* n-gram *is a sequence of words, defined as an* n*-tuple* $(w_1, w_2, \ldots, w_n) \in \mathrm{W}^+$.

From now on, instead of $(w_1, w_2 \ldots, w_n)$, we will write $w_1 w_2 \cdots w_n$ as a shorthand.

Each occurence of an n-gram can be represented by a tuple $(w_1 \cdots w_n, i) \in \mathrm{W}^+ \times \mathbb{N}$, where $i \in \mathbb{N}$ is the position of the n-gram in C. Let S be the set of all n-gram occurences in corpus C, defined as follows:

$$\mathrm{S} = \Big\{ (w_1 \cdots w_n, i) \in \mathrm{W}^+ \times \mathbb{N} : \\ (i \leq k - n + 1) \wedge \\ (1 \leq j \leq n)(w_j = t_{i+j-1}) \Big\}. \tag{2.2}$$

Note that n-grams from S do not cross sentence boundaries set by the punctuation symbols from P. There are exceptions to this rule: when a word and a punctuation following it form an abbreviation, then the punctuation is ignored. The corpus C is preprocessed to reflect this before obtaining n-grams.

2.2 Lemmatisation

Words of an n-gram occur in sentences in inflected forms, resulting in various forms of a single n-gram. In order to conflate these forms to a single n-gram, each word has to be *lemmatised*, i.e., a lemma for a given inflected form has to be found. The context of the word is not taken into account, which sometimes leads to ambiguous lemmatisation. Let $lm : \mathrm{W} \to \wp(\mathrm{W})$ be the lemmatisation function mapping each word into a set of ambiguous lemmas, where $\wp$ is the powerset operator. If a word $w \in \mathrm{W}$ cannot be lemmatised for any reason, then $lm(w) = w$.

Another linguistic information obtained by lemmatisation is the word's part-of-speech (POS). In this work, the following four parts-of-speech are considered: nouns (N), adjectives (A), verbs (V) and stopwords (X). Of these four, stopwords deserve some additional attention. Stopwords are words that appear very frequently in written or spoken natural language communication, so they are sometimes regarded as signal noise in the channel (when viewed through Shannon's model of information [44]). In many text mining applications, stopwords are first filtered out before doing any other text processing. Here, stopwords include prepositions, conjunctions, numbers, and pronouns. Let $POS = \{\text{'N'}, \text{'A'}, \text{'V'}, \text{'X'}\}$ be the set of corresponding POS tags. Let function $pos : \mathrm{W} \to \wp(POS)$ associate to each word a set of ambiguous POS tags. If word $w \in \mathrm{W}$ cannot be lemmatised, then its POS is unknown and is set to POS, i.e., $pos(w) = POS$. Let $POS^+ = \bigcup_{n=1}^{\infty} POS^n$ be the set all POS tag sequences, called *POS patterns.*

2.3 Counting and POS Filtering

Let $f : \mathrm{W}^{+} \rightarrow \mathbb{N}_0$ be a function associating to each n-gram its frequency in the corpus C. It is defined as follows:

$$f(w_1 \cdots w_n) = \left| \left\{ (w'_1 \cdots w'_n, i) \in \mathrm{S} : \right.\right. \\ \left.\left. (1 \leq j \leq n)(lm(w_j) \cap lm(w'_j) \neq \emptyset) \right\} \right|. \tag{2.3}$$

Due to lemmatisation, the obtained frequency is insensitive to n-gram inflection.

Only n-grams of the appropriate POS patterns will be considered collocation candidates. Therefore, there is a need for a function that filters out all n-grams that do not conform to those patterns.

Definition 2.2. *Let $POS_f \subseteq POS^{+}$ be the set of allowable POS patterns defining the* POS filter. *An n-gram $w_1 w_2 \cdots w_n$ is said to pass the POS filter iff:*

$$POS_f \cap \prod_{j=1}^{n} pos(w_j) \neq \emptyset, \tag{2.4}$$

where Π denotes the Cartesian product.

CHAPTER 3

Association Measures

You shall know a word by the company it keeps

John Firth

Association measures (AMs) are used to indicate the strength of association of two words. Note that we say "two words" because all AMs are originally defined for digrams [32], so all existing measures for n-grams where $n > 2$ are basically proposed extensions of digram measures. Choosing the appropriate association measure is crucial to the whole process of extracting collocations, because we use this measure to say whether or not an n-gram is a collocation.

The work done on proposing various AMs and on comparing them be presented in section 3.1, after which the basic definitions for some of them will be given in section 3.2. Section 3.3 gives a formalisation of the process of extending AMs, while some heuristic ways of extending AMs are proposed in the last section.

3.1 Introduction

Association measures used in the literature can roughly be divided into four categories:

- Sorting by pure frequencies—this is the most simple measure where each n-gram gets a score equal to its frequency in the corpus
- Hypotesis testing measures—these are measures that test the *null hypotesis* which states that there is no association between the words beyond chance occurences. They work by computing the probability p

that the event would occur if H_0 were true, and then reject H_0 if p is too low (using a certain significance level). Most commonly used hypotesis testing measures are t-test, likelihood ratios, and Pearson's chi-square test.

- Information theoretic measures—a typical representative of this class is the mutual information measure. Mutual information tells us how much does the information we have about the occurence of one word at position $i+1$ increase if we are provided with information about the occurence of another word at position i.

- Heuristic measures—various authors have tried to define their own measures of collocation strength, and a lot of measures have been taken from other fields, such as biology. Neither of these have a strong formal background, but they all express the idea that two words are more likely to be collocations the more they appear together, and the less they appear without each other. Examples of these measures are the Kulczinsky coefficient, the Ochiai coefficient, the Fager and McGowan coefficient, the Dice coefficient, the Yule coefficient, etc. For a more comprehensive list of these measures along with their formulas, interested reader should refer to [38, pp. 170–171].

A very comprehensive list of 84 association measures can be found in [40]. There are also some interesting ways of extracting collocations using more than AMs and POS information. Pearce [39] uses the fact that collocations are non-compositional so he takes advantage of synonym information from WordNet to see if a candidate digram satisfies this property. For example, from the digram *emotional baggage* he constructs the digram *emotional luggage* substituting *baggage* with its synonym *luggage*. He proceeds to count the number of times this new digram occurs in the corpus, and if there is no significant difference between the occurence of the two variants, then the digram cannot be a collocation as it is obviously compositional. Another interesting way of extracting collocations is given in [40]. There, the author tries to combine results of several AMs in order to judge if an n-gram is a collocation. Values of different AMs are seen as features of each n-gram, and together with a set of manually extracted collocations and non-collocations (the training set), the task of extracting collocations becomes the task of classifications into two classes using some machine learning algorithm.

When comparing AMs, we first have to decide on which measures to put on the test. For example, Evert and Krenn [17] compared t-score, frequency, log-likelihood and chi-square, while Thanopoulos et al. [52] compared t-score, mutual information, chi-square and log-likelihood. In this thesis comparison following measures were used: frequency, mutual information, log-likelihood, chi-square and Dice coefficient. The reason for including Dice coefficient while leaving out t-score lies in the fact that t-score is very similar in nature to log-likelihood and chi-square (it is a hypotesis testing measure), while the Dice coefficient is one of many proposed heuristic measures which have no formal background, and has been found to work well in some cases (for example in retrieving bilingual word pairs from a parallel corpus, see [33]).

Definitions for the mentioned measures will now be given, along with some of their properties.

3.2 Definitions for Digrams

In this section, association measures found in the literature will be described. All of them are defined for digrams.

Pointwise mutual information[*] (PMI) [8] is a measure that comes from the field of information theory, and is given by the formula:

$$\mathrm{I}(x,y)=\log_2 \frac{\mathrm{P}(xy)}{\mathrm{P}(x)\mathrm{P}(y)}, \tag{3.1}$$

where x and y are words and $\mathrm{P}(x)$, $\mathrm{P}(y)$, $\mathrm{P}(xy)$ are probabilities of occurence of words x, y, and digram xy, respectively. Those probabilities are approximated by relative frequencies of the words or digrams in the corpus. Since PMI favors rare events (see, for example, [32, §5.4]), sometimes the following formula is used:

$$\mathrm{I}'(x,y)=\log_2 \frac{f(xy)\mathrm{P}(xy)}{\mathrm{P}(x)\mathrm{P}(y)}. \tag{3.2}$$

Introducing a bias toward more frequent word pairs usually shows better performance than using (3.1) (see, for example [54], [38, pp. 171–172]). For other work on PMI, see [9–11, 51].

The *Dice coefficient* [14] is defined as:

$$DICE(x,y)=\frac{2f(xy)}{f(x)+f(y)}, \tag{3.3}$$

where $f(x)$, $f(y)$, $f(xy)$ are frequencies of words x, y and digram xy, respectivley. The Dice coefficient is sometimes considered superior to information theoretic measures, especially in translating using a bilingual aligned corpus [32].

[*]The definition of mutual information used here is more common in corpus linguistic than in information theory, where the definition of average mutual information is usually used.

The *chi-square measure* is defined as:

$$\chi^2 = \sum_{i,j} \frac{(\mathrm{O}_{ij} - \mathrm{E}_{ij})^2}{\mathrm{E}_{ij}}, \qquad (3.4)$$

where O_{ij} and E_{ij} are observed and expected frequencies in a contingency table [32].

The *log-likelihood* ratio (LL) [38] (entropy version) is defined as:

$$\mathrm{G}^2 = \sum_{i,j} \mathrm{O}_{ij} \log \frac{\mathrm{O}_{ij}}{\mathrm{E}_{ij}}. \qquad (3.5)$$

Log-likelihood is a widely used measure for extracting collocations, often giving very good results. Dunning [15] introduced the measure, using it for detecting composite terms and for the determination of domain-specific terms. McInees [34] gives many possible ways of extending this measure and comapares them. Log-likelihood is often used in exatracting collocations, see for example [17] and [52].

3.3 Extending Association Measures

In the previous section basic AMs for extracting collocations were defined. Since all of them are defined for digrams, AMs need to be extended in some way to make them suitable for extracting trigrams and tetragrams (or even generalize the measures for extracting arbitrary long collocations).

Although some work on extending AMs has been done (see, for example, [4, 34, 45]), so far authors have either concentrated on extending only one measure in more ways or on extending more measures, but in the same way. For example, da Silva and Lopes [45] use *fair dispersion point normalization* as a method of extending ϕ^2, log-likelihood, Dice coefficient, and PMI, but this technique only treats n-grams of $n > 2$ as pseudo-digrams. Their idea is to break the n-gram into two parts, thus treating it as a pseudo-digram. However, there is no single break point—the n-gram is broken on every possible breaking point and the measure is then computed for each of these combinations. The average of all these values is then taken as the value of the chosen AM for the particular n-gram.

On the other hand, McIness [34] uses several models for extracing n-grams, but applies them only to log-likelihood.

A totally different approach was used by Kita et al. [27]. They used their *cost criterion* which depends both on the absolute frequency of collocations and on their length in words. For a candidate n-gram a, they define the reduced cost of a, denoted by $\mathrm{K}(a)$, as:

$$\mathrm{K}(a) = (|a| - 1)(f(a) - f(b)),$$

where b is a *n+1*-gram which a is a subset of (e.g., a could be "in spite" and b could be "in spite of") and $f(.)$ is the frequency of the given n-gram. The collocation candidate a starts as a digram and is then expanded by appending new words. The n-gram a' for which $\mathrm{K}(a')$ has the highest value is then taken as a collocation. For example, we could start with the digram "in spite". That is then expanded with the word "of", which almost always follows it, yielding a greater reduced cost than the initial digram. Trigram "in spite of" is then expanded by, e.g., the word "everything" which was found to follow it sometimes. Since the freqency of "in spite of everything" is rather low as "in spite of" can be followed by a number of fairly likely possibilities, the reduced cost function has its greatest value for "in spite of", indicating that this is the most likely candidate for a full collocation. This approach will not be covered here.

In order to compare the extensions, a formal framework for extensions of AMs will first be given.

Definition 3.1. *Let* W^+ *be the set of all n-grams, and* $\mathscr{F}$ *a set of AMs for digrams defiend as* $\mathscr{F} = \{g | g : \mathrm{W}^2 \rightarrow \mathbb{R}\}$, *where g is a function that takes a digram as an argument and returns a real number. An* extension pattern (EP)* *is a function* G *which takes as arguments an AM g, n-gram length, and an n-gram* $w_1 \cdots w_n$ *and returns the value of the extension of g for n-gram* $w_1 \cdots w_n$*:*

$$\mathrm{G} : \mathscr{F} \times \mathbb{N} \times \mathrm{W}^+ \rightarrow \mathbb{R}, \tag{3.6}$$

where $\mathbb{N}$ *is the set of natural numbers.*

When defining how the value of the extension of g is computed, g_i will be used to denote the natural extension of g for an n-gram of length i. The natural extension g_i is a function that takes i arguments and returns a real number, i.e., $g_i : \mathrm{W}^i \rightarrow \mathbb{R}$. Note that even though $g_2 = g$, g will be used on the left side of the equations, and g_2 will be used on the right hand side. Natural extensions of PMI and Dice coefficient are as follows:

$$\mathrm{I}_n(w_1, \ldots, w_n) = \log_2 \frac{\mathrm{P}(w_1 \cdots w_n)}{\prod_{i=1}^{n} \mathrm{P}(w_i)}, \tag{3.7}$$

*Note that this is just a fancy name for extension of an AM.

$$\mathrm{DICE}_n(w_1,\ldots,w_n)=\frac{nf(w_1\cdots w_n)}{\sum_{i=1}^{n}f(w_i)}, \tag{3.8}$$

where P(.) and f(.) have the same meaning as in the previous section.

Since log-likelihood and chi-square work with contingency tables, their formula for natural extension remains unchanged for n-grams of any length, only the dimensions of the table change.

In terms of definition 3.1, da Silva's fair dispersion point normalization for a tetragram could be written as:

$$\mathrm{G}(g,4,w_1w_2w_3w_4)=\frac{g_2(w_1,w_2w_3w_4)+g_2(w_1w_2,w_3w_4)+g_2(w_1w_2w_3,w_4)}{3}$$

Since theoretically there are infinitely many possible EPs, we have to decide on a subset of them to use with the given AMs. Following is a list of EPs used here for extracting trigrams and tetragrams. The list was made from extensions already found in literature and from some new EPs suggested here for the first time. Not that the subscript of G in the following equations does not have the same function as the subscript in g—this subscript is used only to enumerate the different patterns, not to indicate how many arguments G takes.

$$\mathrm{G}_1(g,n,w_1\cdots w_n)=g_n(w_1,\ldots,w_n) \tag{3.9}$$

It is obvious that G_1 is nothing more than the natural extension of the AM, treating all words in an n-gram equally.

$$\mathrm{G}_2(g,n,w_1\cdots w_n)=\frac{g_2(w_1,w_2\cdots w_n)+g_2(w_1\cdots w_{n-1},w_n)}{2} \tag{3.10}$$

Pattern two computes the average of the strength of the inital word and final $(n-1)$-gram, and initial $(n-1)$-gram and final word. This is just one of the ways an n-gram can be broken into a digram. For example, in the tetragram *weapon of mass destruction*, this pattern would observe how strongly *weapon* and *of mass destruction* are correlated, and how strongly *weapon of mass* and *destruction* are correlated. The rationale behind this pattern is that at least one of the two word-trigram combinations should be strongly associated, giving the tetragram a high score. In this example, the trigram *weapon of mass* will almost always be followed by the word *destruction* in the corpus, giving the tetragram a high score. However, the word *weapon* appears with many other word (and hence, trigrams), so association of *weapon* and *of mass destruction* is very weak. This pattern was used by Tadić and Šojat [51].

$$\mathrm{G}_3(g,n,w_1\cdots w_n)=\frac{g_2(w_1\cdots w_{\lfloor n/2\rfloor},w_{\lceil n/2\rceil}\cdots w_n)+g_2(w_1\cdots w_{\lfloor n/2+1\rfloor},w_{\lceil n/2+1\rceil}\cdots w_n)}{2} \tag{3.11}$$

Pattern three also tries to break up the n-gram into a digram, only in the middle. For example, *weapon of mass destruction* is broken into *weapon of* and *mass destruction*. Comparing patterns two and three with da Silva's fair dispersion point normalization, it is obvious that these two patterns are just some of the addends in his formula.

$$\mathrm{G}_4(g,n,w_1\cdots w_n)=\frac{1}{n-1}\sum_{i=1}^{n-1}g_2(w_i,w_{i+1}) \tag{3.12}$$

Pattern four is interesting in that it is not concerned with the n-gram as a whole, but it rather tries to compute the strength of each digram that is a substring of the n-gram in question and guess the strength of the n-gram based on that. For example, to compute the strength of the tetragram *weapon of mass destruction*, this pattern would compute the strength of the digrams *weapon of*, *of mass*, and *mass destruction*. This example also shows us the greatest weakness of this pattern—some of the digrams constituating the n-gram need not be collocations for themselves, so they will normally recive a low score (*weapon of* and *of mass* in this example), reducing the score for the whole n-gram.

$$\mathrm{G}_5(g,n,w_1\cdots w_n)=g_2(w_1\cdots w_{n-1},w_2\cdots w_n) \tag{3.13}$$

Pattern five looks at the inital and final $(n-1)$-gram in the n-gram. In this example, that means it would look at the strength of association between *weapon of mass* and *of mass destruction*.

$$\mathrm{G}_6(g,n,w_1\cdots w_n)=\frac{1}{\binom{n}{2}}\sum_{i=1}^{n}\sum_{j>i}^{n}g_2(w_i,w_j) \tag{3.14}$$

Pattern six was used by Boulis [5]. It is similar to pattern four, only difference is that this pattern takes all possible word pairings (respecting the order of words) that appear in the n-gram. That means that this pattern would also look at the digrams *weapon mass*, *weapon destruction*, and *of destruction* in addition to those already mentioned for pattern four.

$$\mathrm{G}_7(g,n,w_1\cdots w_n)=g_{n-1}(w_1w_2,w_2w_3,\ldots,w_{n-1}w_n) \tag{3.15}$$

Finally, pattern seven treats an n-gram as an $(n-1)$-gram consisting of all consecutive digrams. That means that in *weapon of mass destruction* the digrams *weapon of*, *of mass*, and *mass destruction* are treated as parts of the trigram whose frequency in the corpus is the frequency of *weapon of mass destruction*, while the frequencies of the words of this new trigram are the frequencies of the mentioned digrams. This pattern is first suggested here.

It is also interesting to note that the presented way of extending n-grams is in some ways very smilar to the work done in [34, §4.1]. For example, pattern one corresponds to her model 1, pattern two is a combination of models 7 and 13, and pattern three corresponds to model 2.

When applying these patterns to trigrams, we get the following instances:

$$\mathrm{G}_1(g,3,w_1w_2w_3) = g_3(w_1,w_2,w_3) \tag{3.16}$$

$$\mathrm{G}_2(g,3,w_1w_2w_3) = \mathrm{G}_3(g,3,w_1w_2w_3) = \frac{g_2(w_1,w_2w_3)+g_2(w_1w_2,w_3)}{2} \tag{3.17}$$

$$\mathrm{G}_4(g,3,w_1w_2w_3) = \frac{g_2(w_1,w_2)+g_2(w_2,w_3)}{2} \tag{3.18}$$

$$\mathrm{G}_5(g,3,w_1w_2w_3) = \mathrm{G}_7(g,3,w_1w_2w_3) = g_2(w_1w_2,w_2w_3) \tag{3.19}$$

$$\mathrm{G}_6(g,3,w_1w_2w_3) = \frac{g_2(w_1,w_2)+g_2(w_2,w_3)+g_2(w_1,w_3)}{3} \tag{3.20}$$

Note here that for trigrams pattern three has the same instance as pattern two and that pattern seven has the same instance as pattern five.
When applying the patterns to tetragrams, we get the following instances:

$$\mathrm{G}_1(g,4,w_1w_2w_3w_4) = g_4(w_1,w_2,w_3,w_4) \tag{3.21}$$

$$\mathrm{G}_2(g,4,w_1w_2w_3w_4) = \frac{g_2(w_1,w_2w_3w_4)+g_2(w_1w_2w_3,w_4)}{2} \tag{3.22}$$

$$\mathrm{G}_3(g,4,w_1w_2w_3w_4) = g_2(w_1w_2,w_3w_4) \tag{3.23}$$

$$\mathrm{G}_4(g,4,w_1w_2w_3w_4) = \frac{g_2(w_1,w_2)+g_2(w_2,w_3)+g_2(w_3,w_4)}{3} \tag{3.24}$$

$$\mathrm{G}_5(g,4,w_1w_2w_3w_4) = g_2(w_1w_2w_3,w_2w_3w_4) \tag{3.25}$$

$$\mathrm{G}_6(g,4,w_1w_2w_3w_4) = \frac{g_2(w_1,w_2)+g_2(w_2,w_3)+g_2(w_3,w_4)}{6} + \\ + \frac{g_2(w_1,w_3)+g_2(w_1,w_4)+g_2(w_2,w_4)}{6} \tag{3.26}$$

$$\mathrm{G}_7(g,4,w_1w_2w_3w_4) = g_3(w_1w_2,w_2w_3,w_3w_4) \tag{3.27}$$

3.4 Heuristic Patterns

In the previous subsection we defined some general patterns for extending any AM for n-gram of any size. However, these patterns showed poor performance when extracting collocations in which one of the words is a stop word. The reason is obvious—stopwords are very frequent in the corpus so the patterns that treat all the words of an n-gram equally give low scores to n-grams that have such words. To overcome this problem, heuristic patterns for trigrams and tetragrams are proposed here (all based on the intuition that for different types of collocations[*] different patterns should be used). This is also in agreement with the fact that stopwords do not carry any meaning, so they can be viewed as a type of signal noise in communication, making it harder to convey meaning. After filtering out the stopwords, the message becomes clearer. Before giving the formulas for heuristic patterns, it should be noted that that even though stopwords are not taken into account, the frequency of the whole n-gram (including the stopword) is. For example, when we write $g_2(w_1,w_3)$, this means that n-gram $w_1w_2w_3$ is treated as a digram whose word frequencies are frequencies of w_1 and w_3, respectively, but the frequency of this digram is the frequency of trigram $w_1w_2w_3$. The proposed patterns are (as a shorthand, $stop(w)$ will denote "'X' $\in pos(w)$"):

$$\mathrm{H}_1(g,3,w_1w_2w_3) = \begin{cases} \alpha_1 g_2(w_1,w_3) & \text{if } stop(w_2), \\ \alpha_2 g_3(w_1,w_2,w_3) & \text{otherwise.} \end{cases} \tag{3.28}$$

This pattern for trigrams simply ignores the middle word if it is a stopword. For example, in the trigram *board of education*, this pattern would not take into account how often word *of* appears in the trigram, only how often do words *board* and *education* appear together in the trigram *board of education*.

[*]That is, collocations with different POS patterns.

$$\mathrm{H}_1(g,4,w_1w_2w_3w_4)=\begin{cases}\alpha_1 g_3(w_1,w_3,w_4) & \text{if } \mathit{stop}(w_2),\\ \alpha_2 g_3(w_1,w_2,w_4) & \text{if } \mathit{stop}(w_3),\\ \alpha_3 g_4(w_1,w_2,w_3,w_4) & \text{otherwise.}\end{cases} \tag{3.29}$$

Heuristic patter one ignores only the stopwords from the n-gram. For example, in the tetragram *gallery of modern art*, this pattern would look at the strength of association between words *gallery, modern,* and *art*, while in the tetragram *holy city of jerusalem* words *holy, city,* and *jerusalem* would be considered.

$$\mathrm{H}_2(g,4,w_1w_2w_3w_4)=\begin{cases}\alpha_1 g_2(w_3,w_4) & \text{if } \mathit{stop}(w_2),\\ \alpha_2 g_2(w_1,w_2) & \text{if } \mathit{stop}(w_3),\\ \alpha_3 g_4(w_1,w_2,w_3,w_4) & \text{otherwise.}\end{cases} \tag{3.30}$$

Heuristic pattern two not only ignores the stopwords, but, based on where the stopword was, ignores one of the words that is not a stopword. For example, in *zakon o morskom ribarstvu* (*law of fishing on sea*), it would take only the words *morskom* and *ribarstvu* into consideration. The rationale behind this is that the word *zakon* (*law*) is also very common, and carries little information. Also, in the tetragram *gallery of modern art*, *gallery* is left out, as there are many other galleries, so the word is quite common. In the case of *pravni fakultet u zagrebu* (*zagreb law school*), this pattern would take the words *pravni* and *fakultet* into consideration, since *zagreb* is the name of the town, so any other town name (of a town that has a law school) can be put instead of *zagreb*.

$$\mathrm{H}_3(g,4,w_1w_2w_3w_4)=\begin{cases}\alpha_1 g_2(w_1,w_4) & \text{if } \mathit{stop}(w_2),\\ \alpha_2 g_2(w_2,w_4) & \text{if } \mathit{stop}(w_3),\\ \alpha_3 g_4(w_1,w_2,w_3,w_4) & \text{otherwise.}\end{cases} \tag{3.31}$$

This pattern also ignores an additional non-stopword in the tetragram, only that the word to be left out is chosen according to a different argument. For example, in the tetragram *gallery of modern art*, words *gallery* and *art* will be considered. Rationale behind this is that the third word is an adjective, so it can be often substituted with another adjective to form a new collocation. In this example, word *modern* could be replaced with *contemporary* or *fine* to form new, perfectly sound collocations. In the case of the tetragram *fakultet* *elektrotehnike i računarstva* (*faculty of electrical engineering and computing*), only the words *elektrotehnike* and *računarstva* are taken into account. The word *faculty* is ignored, for the same reasons as the word *zakon* was ignored in pattern two. An English example would be *buddhist church of vietnam*, where the word *buddhist* is ignored for the same reasons as word *modern* in *gallery of modern art* (*buddhist* could be, for example, replaced with *catholic* or *hindu* to form new collocations).

$$\mathrm{H}_4(g,4,w_1w_2w_3w_4)=\begin{cases}\alpha_1 g_2(w_1,w_3w_4) & \text{if } \mathit{stop}(w_2),\\ \alpha_2 g_2(w_1w_2,w_4) & \text{if } \mathit{stop}(w_3),\\ \alpha_3 g_4(w_1,w_2,w_3,w_4) & \text{otherwise.}\end{cases} \tag{3.32}$$

Heuristic pattern four takes all non-stopwords into account, but unlike pattern one, it treats adjacent words as digrams. For example, in the already mentioned tetragram *weapon of mass destruction*, this pattern will look how strongly the word *weapon* is associated to the digram *mass destruction*. In the tetragram *holy city of jerusalem*, digram *holy city* and word *jerusalem* would be taken into account. In Croatian, for example, in the tetragram *centar za socijalnu skrb* (*center for social welfare*) word *centar* and digram *socijalnu skrb* would be inspected for proof of strong association, while in the tetragram *nobelova nagrada za mir* (*nobel prize for peace*), digram *nobelova nagrada* and word *mir* would be considered.

The parameters α_1, α_2, α_3, and α_4 are chosen so the maximum of the case function they are multiplying is equal to 1. In short, they are used for normalizing the AM scores for different case functions to make them comparable. For example, α_1 in equation (3.32) could be written as

$$\alpha_1 = \frac{1}{\max\limits_{(w_1w_4)\in \mathrm{W}^2} g(w_1,w_4)},$$

where W is the set of words (see chapter 2 for more).

This way, all the cases of the heuristic pattern are given equal weight—there is no bias toward collocations with or without stop words. The difference between these heuristics is the treatment of the non-stopwords—in the first case they are all treated the same, while in other two cases we try to find two words in the tetragram that bare the most information, i.e., we try to find a digram that best represents the tetragram. Note that in (3.32) and (3.35) digrams before or after the stopword are treated as a single constituent.

When dealing with English, second and third words in a collocation of four words can both be stopwords (e.g., *state of the union*), while this is not possible in Croatian. Therefore, heuristic patterns for tetragrams had to be modified for English in order to deal with this type of collocations. For English, the following patterns were used:

$$H_{2'}(g,4,w_1w_2w_3w_4) = \begin{cases} \alpha_1 g_2(w_1,w_4) & \text{if } \mathit{stop}(w_2) \wedge \mathit{stop}(w_3), \\ \alpha_2 g_2(w_3,w_4) & \text{if } \mathit{stop}(w_3) \wedge \neg \mathit{stop}(w_2), \\ \alpha_3 g_2(w_1,w_2) & \text{if } \mathit{stop}(w_2) \wedge \neg \mathit{stop}(w_3), \\ \alpha_4 g_4(w_1,w_2,w_3,w_4) & \text{otherwise.} \end{cases} \tag{3.33}$$

$$H_{3'}(g,4,w_1w_2w_3w_4) = \begin{cases} \alpha_1 g_2(w_1,w_4) & \text{if } \mathit{stop}(w_2) \wedge \mathit{stop}(w_3), \\ \alpha_2 g_2(w_1,w_4) & \text{if } \mathit{stop}(w_2) \wedge \neg \mathit{stop}(w_3), \\ \alpha_3 g_2(w_2,w_4) & \text{if } \mathit{stop}(w_3) \wedge \neg \mathit{stop}(w_2), \\ \alpha_4 g_4(w_1,w_2,w_3,w_4) & \text{otherwise.} \end{cases} \tag{3.34}$$

$$H_{4'}(g,4,w_1w_2w_3w_4) = \begin{cases} \alpha_1 g_2(w_1,w_4) & \text{if } \mathit{stop}(w_2) \wedge \mathit{stop}(w_3), \\ \alpha_2 g_2(w_1,w_3w_4) & \text{if } \mathit{stop}(w_2) \wedge \neg \mathit{stop}(w_3), \\ \alpha_3 g_2(w_1w_2,w_4) & \text{if } \mathit{stop}(w_3) \wedge \neg \mathit{stop}(w_2), \\ \alpha_4 g_4(w_1,w_2,w_3,w_4) & \text{otherwise.} \end{cases} \tag{3.35}$$

Chapter 4

Evaluation

Picture is worth a thousand words

NN

In this chapter the approach used for evaluating the performance of a particular AM-EP combination will be described. This will enable the comparison of not only different AMs, but also different EPs and their (in)dependance of AMs.

First, in section 4.1 the corpora on which the performance is evaluated will be described. Section 4.2 will introduce random samples and describe how they are used to evaluate the performance of AMs, along with all the problems and advantages this kind of evaluation carries. Last section presents the algorithm used to obtain the numerical results that are shown as graphs in the next chapter.

4.1 Corpora

Four text corpora were used for the task of collocation extraction: Vjesnik, Narodne novine, Hrcak and Time. The first three are in Croatian language while the last one is in English. Following is a brief description of each corpus, while a basic statistics for all of them is given in tables 4.2 and 4.1.

Vjesnik [56] is a corpus of Croatian newspaper articles. The particular subset of Vjesnik used here is a part of Croatian National Corpus [50], It comprised of articles from different topics (culture, sports, daily news, economy, local news and foreign affairs), all published between 2000 and 2003. This corpus was chosen as a typical representative of a newspaper corpus.

Narodne novine [37] are an official gazette of the Republic of Croatia. This is a corpus of legal documents—various laws, legal acts, etc. The documents in the corpus were written by the parliament of Republic of Croatia and are thus good representatives of legislative writing style.

Another corpus in Croatian language is **Hrcak**—corpus of scientific texts in Croatian. The texts from Hrcak corpus can be obtained from [24]. The documents in the corpus are all from different scientific journals (from different areas of research) and represent typical scientific writing style.

For a corpus in English, articles from the journal **Time** [53] were chosen. This corpus is intended to be the english counterpart of Vjesnik—all the downloaded articles are from different topics which are very similar to those in Vjesnik.

Note here that the three Croatian corpora differ in writing styles, not only in their domain (the fact that they differ in domain is a side-effect caused by the fact that we were searching corpora with different writing styles). Writing style denotes the structure of sentences in documents (their length, complexity, presence of adjectives and adverbs in them, etc.), average document lengths, repetitive use of words from a restricted vocabulary, etc. The three mentioned writing styles (journalistic, legislative and scientific) have the following characteristics:

Journalistic writing style uses short, simple sentences while trying not to reuse the same vocabulary, but use synonyms instead. The documents are short and without a formal structure and adjectives and adverbs are used moderately.

Legislative style has a very strict document and sentence structure and abstains from using adverbs and adjectives. Vocabulary in these documents is kept at a minimum (cf. table 4.1—Narodne novine corpus has the least unigram types of all three Croatian corpora) and documents range from very short to very long ones.

Scientific style is characterized by long, complex sentences without a very formal structure. The vocabulary is rich as there are many scientific terms from different fields present (cf. table 4.1—Hrcak corpus has the most unigram types of all three Croatian corpora). Adjectives and adverbs are not used much and documents tend to be long.

Beside finding the best combination of AM and EP for each corpus, three important questions will try to be answered:

TABLE 4.1: Basic type statistics

	documents	unigrams	digrams	trigrams	tetragrams*
Vjesnik	40299	210 782	3 524 733	4 653 020	1 779 486
Narodne novine	7008	167 911	1 816 120	4 656 012	1 601 788
Hrcak	3469	512 806	3 920 105	7 450 716	2 496 401
Time	33527	199 826	3 029 454	8 385 518	1 916 748

TABLE 4.2: Basic token statistics

	unigrams	digrams	trigrams	tetragrams*
Vjesnik	18 103 261	16 108 992	9 683 327	2 124 441
Narodne novine	17 417 156	15 414 673	14 038 809	6 163 831
Hrcak	14 623 218	12 367 069	10 837 252	2 999 252
Time	18 525 401	16 696 071	15 057 799	2 181 082

1. Do EPs depend on the AM, or are there some EPs that are generally better than others and that should be used whenever extracting collocations of more than two words?

2. Is the ranking of AMs independant of the writing style, i.e., do some AMs show to perform better than others independent of a style in which the corpus was written, but within the same language? To answer this question, results for the first three corpuses which are all in Croatian, but have different writing styles, will be compared.

3. Is the ranking of AMs language-independent, i.e., do some AMs show to perform better than others independent of a language? To answer this question, results for the Vjesnik and Time corpora will be compared, as they have the same writing style but are in different languages.

The last two questions can also be raised for the ranking of EPs, so they will be addressed as well.

An explanation why it was decided to compare different writing styles and not different domains is in order. This was done because there is much more diversity (with regard to collocation extraction) between different writing styles than between different domains with the same style. Since AMs don't care about the actual meaning of the words but rather their frequencies in the corpus, if one would take for example a collection of maritime laws and a collection of civil laws, AMs would perform very similar on both corpora as they

*Only the statistics for the tetragrams that passed the POS filter are shown.

have the same sentence structure and thus word and n-gram frequencies are distributed similarly. This point is illustrated in tables 4.4a–4.4d. These tables give the Zipf distribution for n-grams in all four corpora, i.e., they give the information about how many n-grams (types) appear a certain number of times in the corpus. The fact that the number of n-grams that appear k times in the corpus drops as k increases is known as Zipf's law [60]. From the tables, one should observe the difference between the first three rows (three corpora of different writing styles), and compare it with the difference between the last two rows (two corpora of the same writing style). Note that all four corpora have almost the same number of tokens. From tables 4.4a–4.4d it is obvious that different writing styles have very different Zipf distributions, while the two newspaper corpora have almost the same distributions, even though they are in different languages. This confirms the characteristics of writing styles given earlier in this section (e.g., the claim that legislative writing style has a more controlled vocabulary and that scientific is quite opposite with a much richer vocabulary due to technical terms from various fields). Still, the claim that writing styles make more difference in collocation extraction than domains do is somewhat based on intuition. However, in the work done by Johansson [26], he compared, among other things, the overlap of digrams between four different genres.* What Johansson found was that there is very little overlap between the digrams extracted from the four different genres. In other words, different genres yield very different collocations. On the other hand, there is no work, known to the author, that deals with collocation extraction from corpora of same genres and different domains. Based on intuition and empirical results from both the corpora used here and the work done by Johansson, and due to lack of literature that would back up the claim that different domains matter in collocation extraction, the assumption that writing styles should be compared (and not domains), is a rational one.

4.2 Sample

In section 1.2 on page 4 an overview of some of the approaches taken by authors to evaluate the results of thier collocation extraction systems was given. Some advantages and disadvantages for each approach were also pointed out. When deciding how to evaluate the results, the following had to be taken into consideration:

*The term *genre* is basically the same as *writing style* used here. Four genres compared by Johansson were press reportage, biographies and memoires, scientifical and technical writing, and adventure and Western fiction novels.

TABLE 4.3: Zipf distribution of n-grams in the four corpora. Each entry in the table shows how many types of n-grams are there in a given corpus, that have the frequency in the corpus equal to the number in the top row of the same column. The numbers are shown in thousands.

Corpus	1	2	3	4	5	6	7	8	9	≥ 10
NN	73	26	12	9	6	4	3	3	2	36
Hrcak	279	77	34	21	13	10	7	6	5	61
Vjesnik	118	32	10	5	3	2	2	1	1	36
Time	111	23	10	6	4	3	3	2	2	34

(a) Unigrams

Corpus	1	2	3	4	5	6	7	8	9	≥ 10
NN	931	312	138	90	53	41	29	23	18	182
Hrcak	2761	525	193	106	64	45	32	24	19	152
Vjesnik	2351	489	194	107	68	47	34	27	22	188
Time	2021	397	162	90	59	42	31	24	19	185

(b) Digrams

Corpus	1	2	3	4	5	6	7	8	9	≥ 10
NN	2937	813	297	176	88	67	43	34	25	176
Hrcak	6325	689	177	83	44	28	19	14	11	61
Vjesnik	7399	847	239	112	64	41	29	21	16	99
Time	6881	810	247	122	71	48	33	25	19	129

(c) Trigrams

Corpus	1	2	3	4	5	6	7	8	9	≥ 10
NN	1123	259	83	46	20	16	9	7	5	32
Hrcak	2260	168	30	13	6	4	2	2	2	8
Vjesnik	1632	105	19	8	4	3	2	1	1	5
Time	1792	94	14	6	3	2	1	1	1	3

(d) Tetragrams

1. We are dealing with four different corpora.
2. The corpora are not standardized, i.e., there is no established set of collocations with which one can compare the results.
3. For each corpus, we are using five different AMs.
4. For each AM, we are extracting collocations consisting of two, three or four words.
5. For each AM used on trigrams, six different EPs will be tested.
6. For each AM used on tetragrams, eleven different EPs will be tested.
7. The number of n-grams in the corpus depends on n, and varies roughly between one and three million n-grams.

In total, 360 different lists of a few million n-grams each will be generated for evaluation. Having that in mind, some of the mentioned approaches for evaluation can be eliminated. Employing the skills of a professional lexicographer to manually tag the n-grams is obviously out of the question, as it would take years to complete this task, even if the expert would evaluate only the first thousand highest rakning n-grams in each list. Thanopoulos' approach is unusable due to fact number 2 in the previous list. The method used by da Silva and Lopes is also unusable for two reasons. Firstly, their approach is based on a list of n-grams that are extracted as multi-word units (i.e., all the n-grams in that list are claimed to be collocations). This is not the case here as the n-grams in each list are just ranked by their value of AM, but no explicit decision is made wheter or not an n-gram is a collocation. Secondly, the problem of a great number of lists is still present. Extracting even a small sample from each of the 360 lists would take very long to inspect manually. What is left is the approach used by Evert and Krenn [17]. Though not completely precise, his method of evaluation was the only sound solution in this case. This reasoning coincides with the statement from [17] which says that "where it is difficult to generalize evaluation results over different tasks and corpora, and where extensive and time-consuming manual inspection of the candidate data is required—random sample evaluation is an indispensable means to make many more and more specific evaluation experiments possible."

Evert's approach consists of a small random sample of n-grams that are manually annotated by a human expert. After obtaining the sample, he simply compares the n-best list of candidates for each AM against this random sample and computes the precision and recall. However, this approach had to be modified somewhat to meet the particular needs of the work done here.

The reason was that we are, unlike Evert, interested not only in precision, but also in recall. Recall is very important as the purpose of collocation extraction presented here was motivated by a need for improvement of a document indexing system. For a document indexing system, it is very important that not to lose any of the potentialy valuable indexing terms. So, in order to get measurable levels of recall, a larger number of positive examples (i.e., true collocations) was needed. If one would simply use a random subset of the corpus, that subset would need to be large in order to find enough collocations in it, as there are normally more non-collocations than collocation (see later in this section for more on this).

That is why the following approach was used: the human expert was presented with a large list of randomly selected n-grams from the corpus and was asked to find 100 collocations. After that, he was asked to find another 100 non-collocations from the same list. This list of 200 n-grams (100 collocations and 100 non-collocations) was then used as a random subset for evaluating the results of each AM. It is important to note here that the list presented to the human expert consisted of only those n-grams that passed the POS filters. The following POS filters were used: AN, NN (for digrams); ANN, AAN, NAN, NNN, NXN (for trigrams); and ANNN, AANN, AAAN, NNNN, NXAN, and ANXN (for tetragrams). For English, the pattern NXXN was also included for tetragrams. It should be noted here that not allowing the first word in an n-gram to be a stopword leads to some decreas of recall. The reason is that the stoplist (list of stopwords) used consisted of prepositions, conjunctions, numbers, and pronouns. Therefore, collocations like *first aid* or *ten commandments* will not pass the POS filter as their POS is XN. However, cases like this account for only a minor part of all the n-grams with that pattern so it was decided that this small loss of recall will be traded for a much greater gain in precision. Recall that for lemmatising and POS tagging, a morphological lexicon constructed by rule-based automatic acquisition [48] was used. The so obtained lexicon is not perfectly accurate, thus prone to lemmatising and POS tagging errors. The words not found in the dictionary were given all possible POS tags. Presenting the human expert with a POS filtered list has two main advantages over presenting him with a non-filtered list. Firstly, since the same POS filter is applied in the actual process of extracting collocations, it is insured that all collocations from the sample will appear also in the list generated by the system. That way we will surely be able to get 100% recall on the sample. The second advantage is that also all negative examples from the sample will appear in the list generated by the system. This is important because otherwise the human expert could generate a sample with a lot of non-collocations that do not pass the POS filter[*], resulting in an unrealistically high precision for all AMs. This high precision would be due to the POS filter and not the AMs, which is obviously not what one would desire.

Using this approach, the human expert had to extract 1200 collocations and 1200 non-collocations (4 corpora × 3 n-gram sizes × 100 n-grams). In comparison, if the human expert was to tag only one hundred higest ranking n-grams for each generated list (recall that this is 360 lists of n-grams), he would need to look at 36 thousand n-grams. Of course, the number of n-grams actually inspected by the human expert was more than 2400, but is still much less than 36 thousand. And this is only the time saved. The real advantage in using the modified method to inspecting first n highest ranking n-grams in a list lies in the fact that the highest-ranking n-grams hardly reflect the real performance of an AM[†], especially when dealing with a few million candidates like in our case.

There are, however, some problems. The lack of a good definition of a collocation (even after deciding on what will be considered a collocation, there is a lot of room left for ambiguity) led to problems with the construction of the sample. For some n-grams it was unclear whether they are collocations or not, even for the human expert. Most of the problems were caused by two types of collocations: technical terms and the *tech and science* collocations. Problem with technical terms is that no human expert can ever be familiar with all the technical terms from all areas of human activity (trazim bolju rijec od activity). This is why the expert is sometimes unsure if some n-gram is a collocation or not—he simply doesn't know enough about the filed in which the n-gram is used to judge about it. Problem with *tech and science* type of collocation is different—it is caused by the fact that this particular type is very hard to define and deciding on whether or not to classify an n-gram as this type of collocation often depends on the subjective point of view of the person performing the classification. In short, problems were caused by two things: lack of knowledge of the human expert (with regard to technical terms) and the vague border between collocations and non-collocations in the *tech and science* type. As an example of vagueness in the *tech and science* type, consider the following n-gram: *mother and child*—to some people that would be a collocation as there is an obvious semantic link between the first and the last word, but again some would argue that this link is not strong enough or that the n-gram simply isn't used as a phrase in speech often enough to be considered a collocation. Whenever the human expert would be in doubt as

[*]See section 5 to see the actual number of n-grams that do not pass the POS filters.

[†]Most of AMs give good collocations at the very top of the list, i.e., they give good precision at the top without giving an indication of what is their recall.

to whether n-gram is or isn't a collocation, he would have discarded it and the n-gram wasn't considered a collocation but it wasn't considered a non-collocation either. These n-grams are said to be *borderline cases*.

Note that the existence of borderline cases does not influence the results of the evaluation as they are never a part of the sample. However, one might argue that we cannot simply ignore their existence and exclude them from the sample altogether. Maybe adding them to the sample either as collocation or as non-collocations would change the final ranking of the AMs? To answer this question, experiments were run with borderline cases all treated as collocations, then all as non-collocations and even with half of borderline cases (randomly selected) treated as collocations and the other half as non-collocations. The total number of borderline cases was 100. This experiment was run only for digrams in *Narodne novine* corpus, as running it on all combinations of corpora and n-gram sizes is out of scope of this thesis. The results showed that there was no change in the ranking of performances of AMs for either of the experiments. This enables the continuation of experiments using the method described above while ignoring the borderline cases and without fear that this would influence the results. The results of the mentioned experiments with included borderline cases will not be shown, in an effort to keep the large amount of the presented material from cluttering the thesis.

One more thing that remains questionable about the method of evaluation used is the fact that a sample with the same number of collocations and non-collocations was used, even though there are more non-collocations than collocations in text (this was mentioned earlier in the section). In short, one might argue that the sample does not reflect the true state of the corpus (which it doesn't) and that therefore the results would be different if a more adequate sample was used (i.e., a sample containing the right ratio of collocations and non-collocations). To see if this is true, this claim was tested on digrams from the *Narodne novine* corpus. The human expert first went through all the digrams from ten randomly selected documents from the corpus and tagged them as collocation, non-collocations or borderline cases. From that list, the total number of collocations and the total number of non-collocations was taken (borderline cases were ignored) and the ratio of non-collocations to collocations was computed. The ratio was found to be 1.3. The reason for this somewhat surprisingly small ratio (one would expect the ratio to be even more in favor of non-collocations) is that the expert was presented with a POS filtered list of n-grams from each selected document. This was done so beacuse the final ranked list of n-grams is also POS filtered, and the ratio of non-collocations to collocations was estimated for this list. After that, the human expert simply found more non-collocations to get a sample with the right ratio and the experiments were run on that sample. The results showed that there is no difference in the ranking of AMs when using a sample with the same number of collocations and non-collocations and this sample (of course, the absolute numbers for precision and recall were different, but we are only interested in the ranking of the AMs). Again, due to the large number of presented material, the actual numbers for this experiment will be omitted.

4.3 Obtaining Results

In the previous section the idea of using a random sample for evaluation was explained. Here it will be shown how that sample is actually used. First, one should decide on a measure for comparing the results. Using the precision-recall graph was decided to be the most appropriate in this case. For every level of recall from 5 to 100%, precision is computed and that represents one point on the graph. It is important to note here that if two or more n-grams had the same value given by an AM, it was randomly decided how they will be ranked. Here is the algorithm for computing the precision and recall:

Algorithm 1: Computing precision and recall

```
positive ← set of positive examples
negative ← set of negative examples
sample ← positive ∪ negative
ngs ← list of all $n$-grams
srtlist ← []
POS ← set of allowable POS patterns
ngs ← filter(lambda x: x ∈ POS, ngs )
for ngram in ngs do
    $i$ ← AM(ngram)
    srtlist.add ((i, ngram))
end
sort(srtlist, order = decreasing)
for $j$ ← 0 to len(srtlist ) − 1 do
    if srtlist[$j$] ∈ sample then
        if srtlist[$j$] ∈ positive then
            np ← np + 1
            if np mod 5 = 0 then
                precision ← $\frac{np}{np+nn}$
                recall ← $\frac{np}{|positive|}$
                points.add ((recall, precision))
            end
        end
        else nn ← nn + 1
    end
end
return points
```

CHAPTER 5

Results

I have had my results for a long time, but I do not yet know how I am to arrive at them.

Karl Friedrich Gauss

In this chapter the results for digrams, trigrams, and tetragrams for all four corpora will be given. As showing the results of all possible combinations of AMs and EPs would take up much space and would only clutter the document, for each AM only the results of its best extension will be shown.

Although a method of evaluating the performance of AMs was chosen, there was nothing said on when one AM is considered to perform better than another. In section 5.1 the criteria for doing this are established. Sections 5.2 to 5.4 on pages 36–40 simply show the results, commenting which measures performed better and which ones performed worse. As results themselves don't mean anything without an interpretation, this is done in the last section of the chapter.

5.1 Comparison Criteria

In chapter 4 the idea of using samples for evaluation of AMs was described. However, it is still unclear how exactly to say whether or not one AM is better than another. Should one AM have better precision than another AM for each level of recall in order to say it performs better? Or does some other criterion exist?

If we recall that these collocations are meant to be used for document indexing, one thing becomes apparent: there is no point in looking at the lower levels of recall, as only the highest levels are of interest. The reason for this is that one would really want to have as much collocations as possible as features for documents—it is not a problem if the system tags some non-collocations as collocations because they will probably be discarded later in the process of feature selection. A much bigger problem is if a collocation is not at all extracted by the system as there is no way to fix this—all the later processing of documents just removes features, it never adds them. That is, if the system fails to identify, for example, “black market” as a collocation, a very valuable information about a document is lost.

For this reason, the recall level of 95% is used as a point on which the comparison will be done. If one AM has higher precision than another on the recall level of 95%, it will be considered better. Why 95% recall? This particular level of recall was chosen instead of the 100% level because there are always collocations in the sample that are very hard to identify using only statistics and POS tagging. The drop of precision from the point of 95% recall to the point of 100% recall is often very big as a consequence of these hard-to-identify collocations. Therefore, taking the precision on the 100% recall for comparison would not reflect the true nature of a measure, as will be shown in the results. On the other hand, the drop of precision from the point of 90% recall to the point of 100% recall is not very noticeable.

There is however, another thing to be careful of. The precision at a particular level of recall does not tell us about how many n-grams are taken as collocations. As it is obvious from algorithm 1, n-grams that are not in the sample are not taken into account when computing precision and recall. Therefore, it is possible for all the n-grams from the sample to be very low ranked in the list of n-grams returned by an AM, but if they are sorted in a way that most of negative examples come after positive ones, the precision will still be high. For example, it would be possible to have a precision of 85% for 95% recall, but in order to get that 95% recall one would have to take 98% of highest-ranking collocations from a list of those that passed the POS filter. In that case, using the AMs looses any meaning as the list of collocations after applying the AM would be almost the same as the list where the AM was not applied. This is why the number of n-grams that are above the last positive example for a particular level of recall is important. Note that when two or more n-grams have the same value of the AM, their order in the list is random.

Example 5.1. *Number of n-grams above the last positive example*
Suppose the sample consists of five positive and five negative examples, and that there are ten thousand n-grams that passed the POS filter (candidates for collocations). Let $p_i, 0 \leq i \leq 5$ denote positive, and $n_i, 0 \leq i \leq 5$ denote the negative examples. All the other n-grams that are not in the sample are not shown here, they are only marked with "$\cdots$". Lines are numbered to indicate the position of an n-gram in the ranked file. Consider the following list:
1 p_3
2 n_1
3 p_4
4 n_2
$\ldots$
4999 p_1
5000 p_2
$\ldots$
9997 n_5
9998 n_3
9999 p_5
10000 n_4

Precision for the 80% recall of the shown list is 4/6, i.e., 66.7%. The number of n-grams that are above the last positive example for 80% recall is 5000 (the last positive example being counted). Now consider the following list:
1 p_3
2 n_1
3 p_4
4 n_2
$\ldots$
9995 p_1
9996 p_2
9997 n_5
9998 n_3
9999 p_5
10000 n_4

In this list, the order of n-grams in the sample is exactly the same as in the previous one. Precision for this list at 80% recall is also 66.7%, but the number of n-grams above the last positive example for 80% recall is much higher—it is 9996. It is obvious that the first list (i.e., the AM that generated this list) is much better than the first one, even though they have the same order of n-grams in the sample.

The previous example showed the problem with using just precision and recall for comparison of AMs. That is why, beside this, another criterion is used for comparison—the already mentioned number of n-grams above the last positive example. For simplicity, this number will be denoted by β, and it will be expressed as the percentage of n-grams (of the total number of n-grams that passed the POS filter) that are above the last positive example for 95% recall. The smaller this number is, the better. Note that β is used as a secondary criterion for comparison. High precision is the first criterion, and the measure that has the highest precision for 95% recall is then inspected using this second criterion to see if the number of n-grams above the 95% recall is very high. If it is, further investigation is needed to declare a best AM for that particular case.

Note here that there was no frequency filter used for obtaining the results. Applying the frequency filter* is a widely used method of trading recall for precision, and because recall was very important here, it was not used.

Sorting the n-grams by pure frequency will not be regarded as an AM, it will be seen more as a baseline performance. There are two reasons for this. First is that frequency is never used alone as a measure for extracting collocations, it is used in the form of a frequency filter to boost precision (see last paragraph). The second reason is that 5–10% of n-grams in each sample appears only once in the entire corpus. Since up to 70% of n-grams in some corpora appear only once, and since n-grams with the same value of an AM are sorted randomly, this means that sometimes even 70% of n-grams obtained by applying frequency are randomly sorted in the list. Obviously, this makes the results obtained by frequency extremly sensitive to the selection of the sample. For all this, frequency was not taken into account as an actual measure.

It is also important to note that I and I' are considered the same measure, and they are denoted by MI and MI' in the graphs (for better readability). The better of the two is always shown, never both of them.

5.2 Digrams

5.2.1 Narodne novine

The results for digrams in the Narodne novine corpus are shown in figure 5.1. Out of 1816120 different digrams in the corpus, 898641 digrams (49.5%) passed the POS filter. From figure 5.1 it is clear that for 95% recall AM with the highest precision is mutual information. MI achieved 79.8% precision, while the next best AM is Dice coefficient with 63.3%. β for mutual information is 42%

*That is, keeping only those n-grams that appear in the corpus more than a certain number of times.

(377711 n-grams), while the AM with the lowest β was Dice whose β was 18.4% (165434 n-grams). Since Dice had more than two times lower β than MI, a third criterion was used to decide which AM is better for this corpus. This criterion is the precision on 100% recall. Dice had a precision of 51.8%, while MI achieved a precision of 58.8% on the 100% recall. Therefore, MI is considered the best AM for digrams on the NN corpus.

5.2.2 Vjesnik

The results for digrams in the Vjesnik corpus are shown in figure 5.2. Out of 3524733 different digrams in the corpus, 1420557 digrams (40.3%) passed the POS filter. Here, the frequency-biased version of mutual information (MI') is the AM with the highest precision, with the value of 81.2%. Second best was Dice with the value of 72.5%. β for MI' is 16.3% (232245 n-grams), which is also the best result. Second lowest β was 21.4% (303299 n-grams) achieved by Dice. Therefore, MI' is the best AM for digrams on the Vjesnik corpus.

5.2.3 Hrcak

The results for digrams in the Hrcak corpus are shown in figure 5.3. Out of 3920105 different digrams in the corpus, 2098817 digrams (53.5%) passed the POS filter. MI' was again found to have the highest precision—71.4%, while the second best was again Dice which closely followed with 69.3%. β for MI' was 39.8% (836131 n-grams), while Dice was the best with a β of 30.3% (636367 n-grams). Neither the difference in precisions nor the difference in β's was enough to say that one measure was better than the other. The precision at 100% recall was again taken as the third criterion. For MI', precision was 55.6%, while for Dice it was also 55.6%. Therefore, the results for Hrcak are inconclusive as either Dice or MI' could be considered to be the best.

5.2.4 Time

The results for digrams in the Time corpus are shown in figure 5.4. Out of 3029454 different digrams in the corpus, 1190293 digrams (39.3%) passed the POS filter. MI' has again showed to have the highest precision—77.8%, while the second best was Dice with 71.9%. MI' had a β of 22.7% (269666 n-grams), while the best was Dice with $\beta = 20.5\%$ (244130 n-grams). Since the difference of β's for MI' and Dice is smaller than the difference in their precisions, MI' is considered to be the best AM for digrams on the Time corpus.

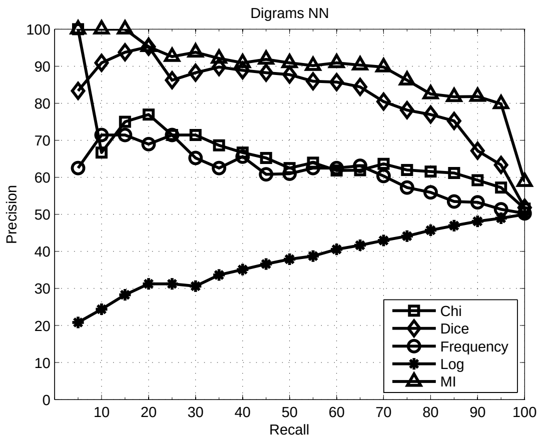


Figure 5.1: Digram results for NN corpus

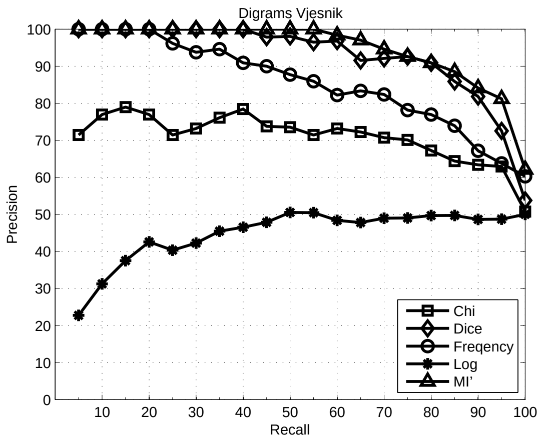


Figure 5.2: Digram results for Vjesnik corpus

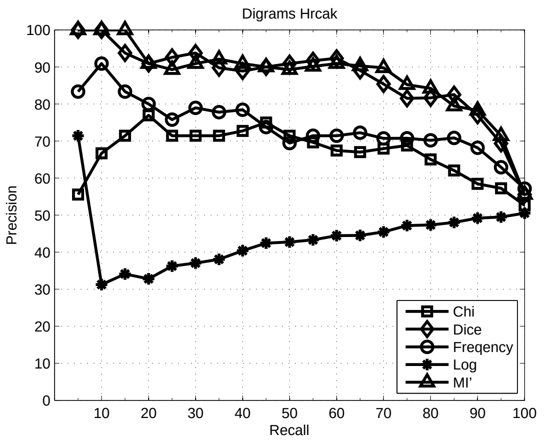


Figure 5.3: Digram results for Hrcak corpus

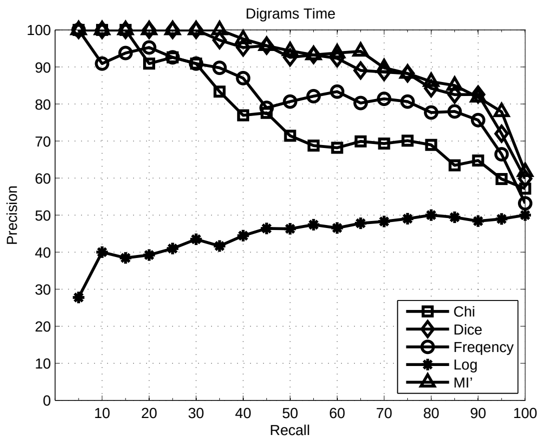


Figure 5.4: Digram results for Time corpus

5.3 Trigrams

5.3.1 Narodne novine

The results for trigrams in the Narodne novine corpus are shown in figure 5.5. Out of 4656012 different trigrams in the corpus, 1570200 trigrams (33.7% of all trigrams) passed the POS filter. The measure with the highest precision was heuristic pattern of mutual information with the precision of 73.1%. Second best measure was G_4 of chi-square with precision of 60.9%. β for MI was 49.9% (784015 n-grams), while the best measure regarding β was heuristic pattern of Dice with β of 32.7% (513054 n-grams). G_4 of chi-square had a β of 61.3% (990558 n-grams), while the precision for heuristic Dice was 58.2%. As G_4 of chi-square has both a lower precision and a higher β, it is clear that it cannot be the best measure. Heuristics pattern of Dice has a 15% lower β, but it also has a 15% lower precision. Since precision is the main criterion for comparison, heuristic pattern of MI is considered the best measure for trigrams on the NN corpus. For log-likelihood, the best extension regarding precision was G_4, and for Dice that was G_2.

5.3.2 Vjesnik

The results for trigrams in the Vjesnik corpus are shown in figure 5.6. Out of 4653020 different trigrams in the corpus, 2161053 trigrams (46.4%) passed the POS filter. Heuristic pattern of MI' was the best measure regarding precision, with the value of 77.2%. Interestingly, pure freqeuncy was the second best with precision of 72.5%. However, due to reasons explained in section 5.1, frequency is not really regarded as an AM. The third best measure was G_2 of Dice with precision of 67.4%. β for heuristic pattern of MI' was 46.5% (1004447 n-grams), while the best measure when comparing β's was heuristic pattern of Dice, with $\beta = 42.2\%$ (911825 n-grams). β for G_2 of Dice was 51.7% (1118300 n-grams). As the difference between β's is 3.3% and difference in precisions was 9.8%, and taking into account that precision is given more wieght, heuristic pattern of MI' is the best measure for trigrams on Vjesnik corpus. For log-likelihood and chi-square, G_4 was found to be the best extension regarding precision.

5.3.3 Hrcak

The results for trigrams in the Hrcak corpus are shown in figure 5.7. Out of 7450716 different trigrams in the corpus, 2834970 trigrams (38.0%) passed the POS filter. Heuristic pattern of MI' was again found to be the best measure with the precision of 70.9% for 95% recall. The second best measure was again frequency with precision of 66.0%, while the third best was G_4 of chi-square with precision of 63.3%. β for heuristic pattern of MI' was 57.0% (1616708 n-grams), while the best measure for β was heuristic pattern of Dice with β of 55.0% (1558648 n-grams). β for G_4 of chi-square was 77.5% (2198437 n-grams), while precision for heuristic pattern of Dice was 61.7%. The difference of 2% in β is not enough for heuristic Dice to make up the difference of 9.2% in precision. Therefore, heuristic pattern of MI' is the best measure for trigrams on Hrcak corpus. For log-likelihood and chi-square, G_4 was again found to be the best extension regarding precision.

5.3.4 Time

The results for trigrams in the Time corpus are shown in figure 5.8. Out of 8385518 different trigrams in the corpus, 1804455 trigrams (21.5%) passed the POS filter. Heuristic pattern of MI' was yet again found to be the best measure with precision of 74.2%. G_4 of Dice and chi-square are at the second best with the same precision of 70.4%. β for heuristic pattern of MI' was 40.9% (738568 n-grams) which was also the best β among all measures, while β for G_4 of Dice was 56.9% (1027133 n-grams) and for G_4 of chi-square was 52.8% (952802 n-grams). Having the best precision and the best β, heuristic pattern of MI' is clearly the best AM for trigrams on Time corpus. G_4 was the best extension of log-likelihood regarding precision.

5.4 Tetragrams

Due to a large number of tetragrams in the corpora, they were filtered by POS pattern during the process of extraction from documents, and not after like digrams and trigrams.* Because of this, it is not possible to give the number of tetragrams that passed the POS filter as a percentage (as the total number of tetragrams is unknown).

5.4.1 Narodne novine

The results for tetragrams in the Narodne novine corpus are shown in figure 5.9. There were 1601789 tetragrams that passed the POS filter. The AM with the highest precision on the 95% recall is H_2 pattern of MI' with 68.3%, while the second best AM was G_1 of chi-square with 62.9%. β for H_2 of MI'

*This is an implementation issue—storing all the n-grams requires too much memory.

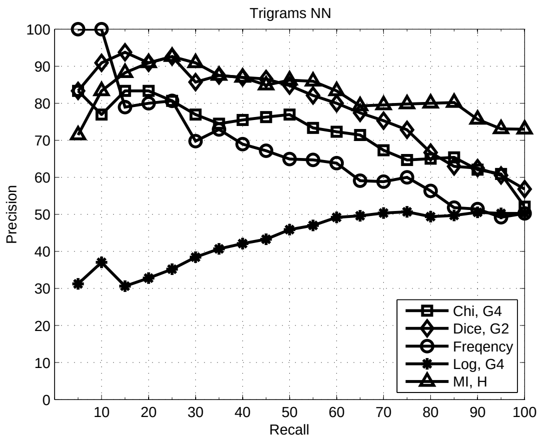


Figure 5.5: Trigram results for NN corpus

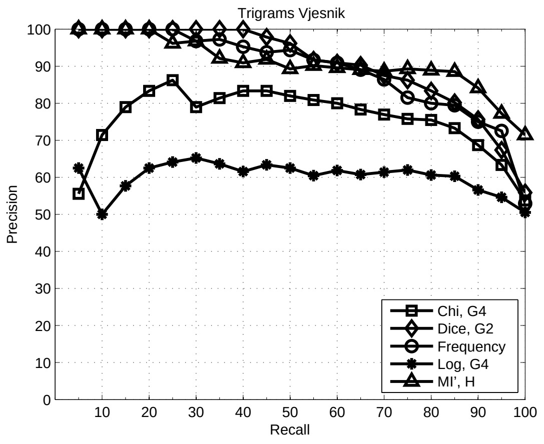


Figure 5.6: Trigram results for Vjesnik corpus

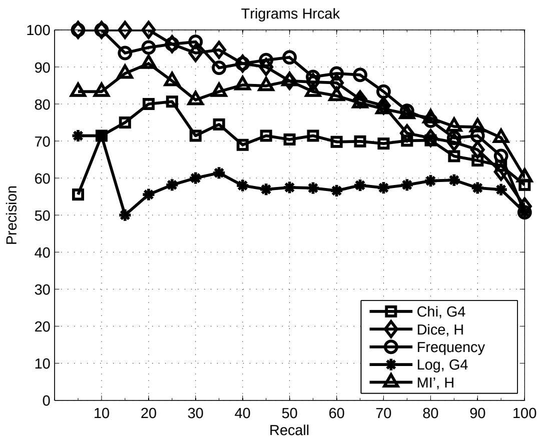


Figure 5.7: Trigram results for Hrcak corpus

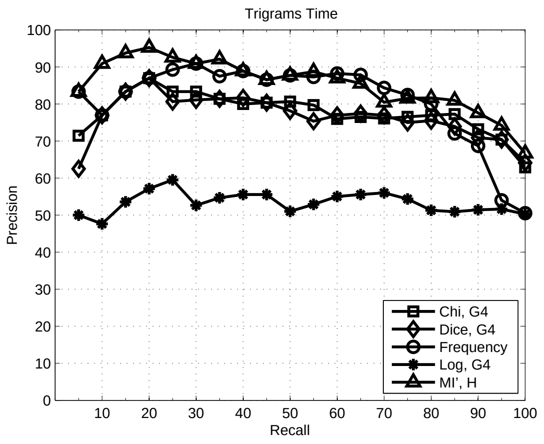


Figure 5.8: Trigram results for Time corpus

was 37.8% (604826 n-grams), while the best AM regarding β was G_5 of log-likelihood with β of 33.9% (543256 n-grams). β for G_1 of chi-square was 55.5% (889771 n-grams), while precision for G_5 of log-likelihood was 54.9%. Clearly, G_1 of chi-square has a lower precision and a higher β, therefore it cannot be the best measure. Log-likelihood's 3.9% lower β does not make up for the difference of 13.4% in the precision that is in favor of MI', hence H_2 pattern of MI' is the best measure for tetragrams on NN corpus. H_1 was the best pattern for Dice, while H_4 was the best pattern for log-likehood regarding precision.

5.4.2 Vjesnik

The results for tetragrams in the Vjesnik corpus are shown in figure 5.10. There were 1779487 tetragrams that passed the POS filter. H_2 pattern of MI' was again found to be the best AM regarding precision, with the precision of 78.5%. G_6 of Dice coefficient was the second best with precision of 76.0%. β for MI' was 56.7% (1008405 n-grams), with the best measure for β being G_6 of Dice with β of 48.5% (863934 n-grams). If the sole criterion for comparison would be precision, MI' would be the best measure. However, β for Dice was 8.2% lower than that of MI', while precision of MI' was only 2.5% higher. The difference in precisions was not even comparable with the difference in β, therefore MI' cannot be declared the best measure for Vjesnik corpus. However, it is still not clear whether Dice should be considered the best measure either—having a lower β is not enough. That is why another, third, criterion was used in this case—precision for the 100% recall. For MI', this value was 57.5%, while for Dice it was 69.9%. Seeing this, G_6 of Dice coefficient was declared the best measure for tetragrams on Vjesnik corpus. For chi-square, G_1 was the best extension pattern, while for log-likelihood that was H_4.

5.4.3 Hrcak

The results for tetragrams in the Hrcak corpus are shown in figure 5.11. There were 2495401 tetragrams that passed the POS filter. Association measure with the highest precision was in this case G_1 of chi-square with precision of 62.1%. Second best was G_6 of MI with precision of 61.3%. β for chi-square was 85.3% (2129467 n-grams), while the AM with the lowest β was G_3 of log-likehood with a β of 59.8% (1492371 n-grams). G_6 of MI had a β of 83.2% (2075994 n-grams). Precision for G_3 of log-likelihood was 50.2%. The difference between chi-square and MI (log-likelihood has a too low precision to be considered as being the best measure) in both precision and β is too small for one to decide on a best measure just by these two criteria. That is why precision for 100% recall was again examined. Chi-square had a precision of 51.0% for 100% recall, while MI had 50.2%. Again, the difference was too small to give any real conclusion, so both of these measures can be considered to be the best measures for tetragrams on Hrcak corpus. However, both of these measures do not perform well in the sense that they require over 80% of all candidates to achieve 95% recall on the sample. This means that more than 5% of collocations from the sample are ranked very low by both of these measures, which a good AM should not do. For this, the results for tetragrams on Hrcak corpus are considered inconclusive. The best extension with regard to precision was H_4 for log-likelihood and G_6 for Dice.

5.4.4 Time

The results for tetragrams in the Time corpus are shown in figure 5.12. There were 1916748 tetragrams that passed the POS filter. The best AM with regard to precision was H_1 pattern of Dice with precision of 61.7%, while the second best was H_1 pattern of MI' with precision of 59.7%. β for Dice was 73.6% (1409972 n-grams), while β for MI' was 73.2% (1403589 n-grams). AM with the lowest β was G_5 of log-likelihood with a β of 61.5% (1178277 n-grams) and a precision of 56.2% (this pattern of log-likelihood was also its best extension pattern regarding precision). Although Dice has a higher precison and a lower β than MI', the difference is not great enough to be able to say Dice is better. Log-likelihood will also not be discarded as a candidate for the best measure because its precison is 5.5% lower than that of Dice which is not a lot given it has a 12.1% lower β. Precision on the 100% recall is inspected for these three measures. Precision for Dice on 100% recall was 52.6%, for MI' it was 50.2%, while it was 52.3% for log-likelihood. Since MI' does not perform best on any of the criteria, it is discarded as a candidate for the best AM. The difference in precision for log-likelihood and Dice was 0.3% which is not a lot so they could both be regarded as performing the best. The difference between the two should be obvious—while Dice will generally give higher precision, it will require more candidates which means that the collocations from the sample are not very high ranked, but rather the non-collocations from the sample are low-ranked. On the other hand, log-likelihood gives a lower precision while requiring less candidates. That means that it gives good scores to both collocations and non-collocations from the sample. The best pattern of chi-square with regard to precision was G_1.

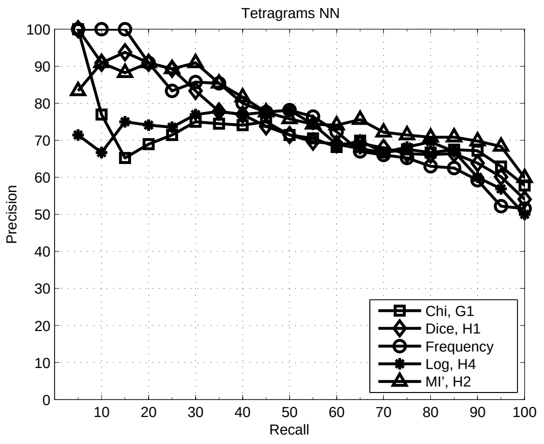


Figure 5.9: Tetragram results for NN corpus

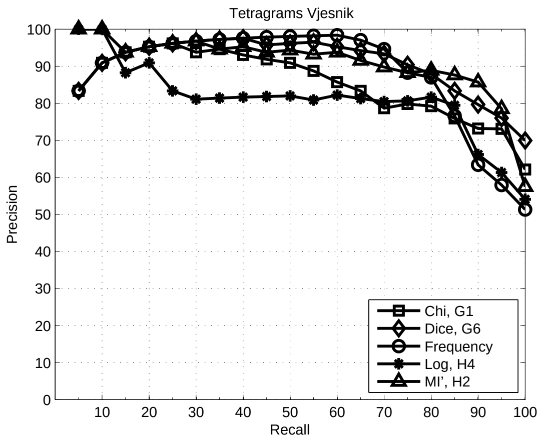


Figure 5.10: Tetragram results for Vjesnik corpus

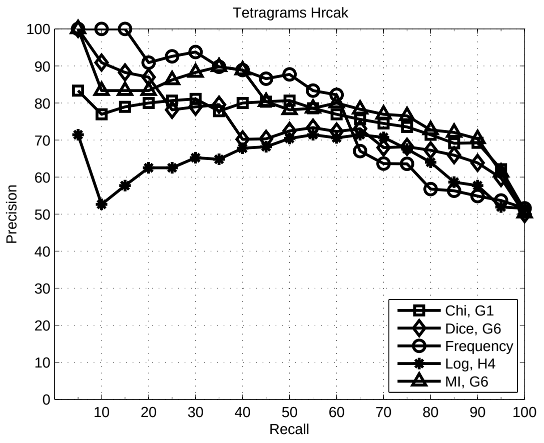


Figure 5.11: Tetragram results for Hrcak corpus

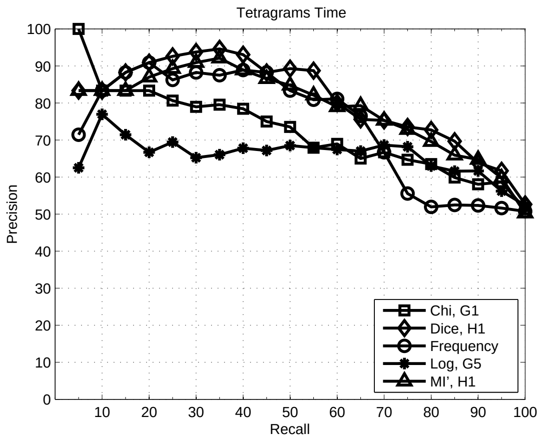


Figure 5.12: Tetragram results for Time corpus

TABLE 5.1: Summary of results. Number before the corpus name indicates digrams, trigrams or tetragrams. If there are two measures that perform equally well, they are both given, with one of them in parenthesis.

Corpus	Best AM	Precision at 95% recall	β at 95% recall
2 NN	MI	79.8	42.0
2 Vjesnik	MI’	81.2	16.3
2 Hrcak	MI’ (Dice)	71.4 (69.3)	39.8 (30.3)
2 Time	MI’	77.8	22.7
3 NN	MI, H	73.1	49.9
3 Vjesnik	MI’, H	77.2	46.5
3 Hrcak	MI’, H	70.9	57.0
3 Time	MI’, H	74.2	40.9
4 NN	MI’, H_2	68.3	37.8
4 Vjesnik	Dice, G_6	76.0	48.5
4 Hrcak	chi-square, G_1 (MI, G_6)	62.1 (61.3)	85.3 (83.2)
4 Time	Dice, H_1 (log-likelihood, G_5)	61.7 (56.2)	73.6 (61.5)

5.5 Discussion of Results

In sections 5.2–5.4 the results of perfomance of various AMs on the four chosen corpora are given. All the most important results from those sections are summarized in table 5.1. Using table 5.1, some of the questions raised at the end of section 4.1 will now be answered.

To see if the order of AMs is independant of writing style, one has to look at the results for the three Croatian copora—NN, Vjesnik, and Hrcak. For digrams, mutual information performed the best on all three corpora, with Dice performing equally good as MI on Hrcak. For trigrams, mutual information performed the best on all three corpora, while for the results for tetragrams were a little different—MI performed the best on NN, Dice was the best on Vjesnik, and chi-square and MI were equally good on Hrcak. From this, it is clear that for digrams and trigrams, mutual information outperforms all the other AMs, independent of the writing style, but within the same language. For tetragrams, the results show that there in no AM that is the best on all three corpora, indicating that as the length of collocations grows, they tend to become more and more corpus-specific*, so one really needs to find an AM that

*For digrams and trigrams, the collocations in the sample were more general, widely used in the language, e.g., *comic book, energy source,video camera*, whereas for tetragrams half of the collocations in the sample were named entities, e.g., *vice president Dick Cheney, Buddhist Church of Vietnam*.

suits the specific corpus when extracting those longer collocations. However, mutual information again showed to perform well, it was the best AM on the NN corpus, and on the Hrcak corpus chi-square and MI performed equally well, outperforming all the other AMs. In short, for digrams and trigrams the results show that there the order of AMs does not depend on the writing style, and that the best measure is mutual information. For tetragrams, the results show that there are variations in the ordering of AMs depending on the writing style.

To see if the order of AMs is indepemdant of languages, the results for Vjesnik and Time corpora should be examined. The results for digrams and trigrams show that mutual information outperformed all other measures on both corpora. For tetragrams, Dice coefficient was the best for both corpora, while for Time log-likelihood performed equally well as Dice. The results are interesting as they show that there is little difference in results between Croatian and English on the same type of corpora. It seems that there is more variation between the length of n-grams than in languages or in writing styles. However, it should be noted here that mutual information was always very close to the best AM, in cases where it was not the best AM, while log-likehood performed worse than all AMs, except for tetragrams in Time corpus. The fact that log-likelihood did not perform well contradicts the claims made in [15], but other authors have also found that log-likehood gives poor results in comparison to other AMs (cf. [45]). The fact that mutual information was found to be the best AM almost always in Croatian corpora coincides with the results obtained by Pecina [40] where 84 AMs were compared and mutual information showed to perform the best. It is very interesting to note that Pecina has done his experiments on a corpus that was in Czech, which is a Slavic language, same as Croatian. Even when he automatically extracted the best subset consisting of 17 measures, MI was among them (while neither log-likelihood nor chi-square were selected). Also, mutual information showed very good performance in [45].

One other question remains—are there some extension patterns that are generally better than others? To see that, the best pattern for each measure is given in table 5.2.

Looking at table 5.2, it is obvious that results for trigrams and tetragrams are very different. For trigrams, pattern four was consistently the best EP for chi-square, same as for log-likelihood. Pattern four expresses the idea that a trigram like “crash test dummy” should be seen as a digram consisting of entities “crash test” and “test dummy”. The best extension for Dice did not show any consistent results—sometimes it was pattern two, sometimes pattern four, and sometimes it was the heuristic pattern. The best pattern for mutual information was always the heuristic pattern.

TABLE 5.2: Best extension patterns for trigrams and tetragrams, with regard to precision.

corpus	Chi-square	Dice coefficient	Log-likelihood	MI
3 NN	G_4	G_2	G_4	H
3 Hrcak	G_4	H	G_4	H
3 Vjesnik	G_4	G_2	G_4	H
3 Time	G_4	G_4	G_4	H
4 NN	G_1	H_1	H_4	H_2
4 Hrcak	G_1	G_6	H_4	G_6
4 Vjesnik	G_1	G_6	H_4	H_2
4 Time	G_1	H_1	G_5	H_1

For tetragrams, the best EP for chi-square was always pattern one which is the natural extension of the measure, saying that all the words in the tetragrams should be treated equally. For Dice, heuristic pattern one was the best two times, and the other two times the best EP was pattern six. For tetragrams, log-likelihood did not show the same results as chi-square, as was the case for trigrams. Heuristic pattern four was the best EP for Croatian corpora, while pattern five was the best for Time corpus. Best patterns for mutual information were heuristic patterns for NN, Vjesnik, and Time corpora, and pattern six for Hrcak corpus.

From this, it is obvious that mutual information benefits greatly from the heuristic patterns, both for trigrams and for tetragrams. Extension patterns for Dice showed a great variation, so no conclusion can be made from those results. For trigrams, log-likelihood favored the same EPs as chi-square, but for tetragrams it too had benefited from heuristic patterns. Chi-square never benefited from the heuristic patterns, which was strange. That is why a more thorough investigation was conducted to find out why this is so. It turned out that the problem with chi-square and heuristic patterns was the following: some POS patterns had very strong collocations,[*] much stronger than all other with the same pattern. Those collocations were then assigned a score (by chi-square) much higher than all other with the same pattern. Meanwhile, other POS patterns did not have those cases. Thus, when computing the α coefficients for each POS pattern, α for the patterns with the very strong collocations was much lower than it should really be, resulting in a low score for all the n-grams with that pattern. To illustrate this problem, consider the following example.

[*]Collocations with a very high value of an AM.

Example 5.2. *Unequal treating of POS patterns*
Consider that we have two classes of POS patterns for trigrams: one in which the second word is a stopword, and the other one which covers all other cases. When computing α_1 and α_2 from equation (3.28), we have to find the trigrams with each of those two patterns that are given the highest score by the AM in question. Imagine the list of trigrams in which the second word is a stopword has the scores (trigrams with this pattern are denoted t_i):

t_1	10432
t_2	178
t_3	176
t_4	176
t_5	175

Now, imagine the list of trigrams which have no stopword has the scores (those trigrams will be denoted u_i):

u_1	576
u_2	556
u_3	555
u_4	553
u_5	552

By dividing the scores of the first list by 10432 (which is equal to multiplying them with $\alpha_1 = 1/10432$) and scores of the second list by 576, and then sorting the list, we get the following:

t_1	1
u_1	1
u_2	0.965
u_3	0.963
u_4	0.960
u_5	0.958
t_2	0.017
t_3	0.016
t_4	0.016
t_5	0.016

It is obvious that the trigrams with the first pattern are being “punished” because there is a very strong collocation with that pattern. This is exactly what is happening with chi-square and why it does not perform well with heuristic patterns.

CHAPTER 6

Conclusion of the First Part

A conclusion is the place where
you got tired of thinking

Arthur Bloch

The first part of this thesis deals with the process of extracting collocations from a large textual corpus. Particular interest is given to the comparison of different association measures, functions used to indicate the strength of bond between the words. The work done here is actually an extension of the work done in [41] where the measures were compared only for digrams and trigrams on the *Narodne Novine* corpus. As the motivation for the work lied in improving a indexing system for documents in Croatian, special interest is given to results obtained on Croatian corpora. The most important contribution of the first part of this thesis is that different association measures were compared for Croatian here for the first time.

After explaining the notion of collocation and all the problems it bares with it, motivation has been given for its use in the field of natural language processing, followed by a formal approach to preprocessing of the corpus.

This is followed by a brief introduction to the issue of association measures and extending them for trigrams and tetragrams. Dealing with extending AMs in the way it was presented in chapter 3 is completely new and is proposed here for the first time. With the framework given there, some of the different approaches proposed by various authors (da Silva and Lopes [45], McIness [34], Boulis [5], and Tadić and Šojat [51]) were succesfully modeled, and some new approaches were also proposed. As stopwords can sometimes be parts of collocations (as is the case here), some ways of overcoming the problem of their very high frequency is also proposed in the form of heuristic extension patterns. This is a generalization of the idea given in [41].

Evaluating the performance of AMs is always problematic, and the specific approach taken here is thoroughly described in chapter 4. Small random samples used here are a refinement of the approach taken by Evert and Krenn [17] to fit the specific needs of this task. While it is not claimed that this approach will lead to accurate values for precision and recall, it was shown that it is sufficient for the purpose of comparing AMs.

The results of comparing AMs on four corpora are given in chapter 5. These results showed that there is no difference in the ranking of AMs when collocations are extracted from corpora whose documents are written in different ways and that there is little difference in ranking of AMs when extracting collocations from corpora in Croatian and English. It was also interesting that mutual information was the best measure in most cases, a result obtained by Pecina [40] on a Czech corpus. The fact that log-likelihood gave very poor results contradicts the work done by Dunning [16]. However, upon inspection of 56 digrams that were ranked highest in [16] (according to their log-likelihood score), only 15 of them would be considered collocations in the sense they were defined in this thesis. For example, the five highest ranked digrams were *the swiss, can be, previous year, mineral water,* and *at the*—only *mineral water* would be considered a collocation. This also shows how important is the definition of collocation that is adopted—some authors would consider *can be* and *at the* to be valid collocations. In such case, their results would certainly differ from those given here. It is also important to note that Dunning only compared log-likelihood to chi-square, he never compared it to mutual information or Dice coefficient.

Different extension patterns were also compared for every tested AM and the results showed that different AMs have different preffered patterns. The heuristic patterns proposed here have shown to give the best results in most cases, which indicates more work should be done on developing extension patterns that treat n-grams differently based on their POS.

Part II

Application in Text Mining

CHAPTER 7

Letter n-grams

They call it golf because all the other four-letter words were taken

Ray Floyd

Beside using words to represent text, it is possible to use other text features. Letter n-grams are one of such features. This chapter explains what exactly are letter n-grams and on what tasks have they been applied. Some of their weak and strong points are also given.

7.1 Introduction

An n-gram is a subsequence of n items from a given sequence. A *letter n-gram* (sometimes also called *character n-gram*) is a sequence of n characters extracted from text. To generate the n-gram vector for a particular document, a window of length n is moved through the text, sliding forward by a fixed number of characters (usually one) at a time. At each position of the window, the sequence of characters in the window is recorded.

For example, the word "technology" has the following tetragrams (sequences of four characters): "tech", "echn", "chno", "hnol", "nolo", "olog", and "logy". However, strings are often padded with one leading and one closing space so each string of length N has $\mathrm{N} - n + 3$ n-grams. In the case of the word "technology", two more tetragrams would be extracted—"␣tec" and "ogy␣", where the character "␣" represents a space. There are also other issues to note when extracting letter n-grams. Sometimes only characters from the alphabet are taken into account, while ignoring digits and other characters like punctuation marks, quote signs, etc. Special care should also be taken when crossing the word boundaries as some systems prefer that n-grams don't cross word boundaries, while others do. In case that n-grams do cross word boundaries, the phrase "full text" would have the following tetragrams: "␣ful", "full", "ull␣", "ll␣t", "l␣te", "␣tex", "text", and "ext␣". In case n-grams do not cross word boundaries, the following letter n-grams would be extracted: "␣ful", "full", "ull␣", "␣tex", "text", and "ext␣".

7.2 Applications of Letter n-grams

Letter n-grams have a wide variety of applications. First work on letter n-grams was done by Shannon [44]. He wanted to know, given a sequence of letters, the likelihood of the next letter. Mathematically speaking, he wanted to know the probability of the letter x_i, given the last N events were $x_{i-1}, x_{i-2}, \ldots, x_{i-\mathrm{N}}$, i.e., he wanted to know $\mathrm{P}(x_i | x_{i-1}, x_{i-2}, \ldots, x_{i-\mathrm{N}})$. He used this Markov model to simulate a source that generates human language (in his case, English).

Dunning [16] uses letter n-grams for identifying human languages. The advantage of using letter n-grams for this purpose is that they require no a priori linguistic knowledge, and they can be applied to any language, whereas using words for this purpose leads to problems when dealing with languages like Chinese or Japanese, as texts in those languages cannot be tokenized into words. The results obtained from [16] are very promising as very short strings of text (of about dozen characters) can be identified using only a small training corpus (around 100 KB).

Letter n-grams are also used in text categorization. Cavnar and Trenkle [6] used profiles of n-gram frequencies to classify Usenet newsgroup articles, achieving around 80% correct classification rate. They have also succesfully used them for identifying languages.

Jalam [25] used letter n-grams for categorization of multilingual texts. His use of letter n-grams is two-fold—he uses them to first identify the language of the text, then translates the text into French and proceeds to classify the text using letter n-grams-based classifiers built for French. This approach enables building classifiers only for one language, instead of building a classifier for each language in which texts need to be classified. To add support for new languages, one needs only to translate the texts, leaving the classifiers unchanged. Similar work has also been done by Damashek [12], and Biskri and Delisle [3].

There are also information retrieval systems based on letter n-grams [35]. The advantages of using letter n-grams over words in IR systems are numerous: letter n-grams are less sensitive to errors (e.g., if the document contains the word “claracter” insted of “character”, these two words still have five letter trigrams in common), using letter n-grams achieves language independence without having to use language-specific stemmers, lists of stopwords, etc., and using longer letter n-grams than span over word boundaries captures relations between pairs of words (e.g., the n-gram “of co” is the first n-gram in the phrase “of course”).

In summary, the main advantage of letter n-grams is that they are very simple to implement, require no a priori linguistic knowledge, and are very robust to noise that is typically present in large textual corpora (especially when text is obtained using OCR). One of the points that will be addressed in this thesis is the appropriatness of letter n-grams as features for documents to be visualized using correspondence analysis.

CHAPTER 8

Correspondence Analysis

Everything has beauty, but not everyone sees it

Confucius

Correspondence analysis is an exploratory technique for visualizing large amounts of data. It is used to visualize the structure within a matrix, where the rows and columns represent two categorical variables. In this thesis, correspondence analysis will be used to visualize a corpus of newpaper articles, which is represented by a matrix with different text features as columns and with documents as rows.

After an introduction to correspondence analysis in section 8.1, section 8.2 lists some of the applications of the method in various fields. The final section goes into detail to describe the mathematical background of correspondence analysis.

8.1 Introduction

The term *correspondence analysis* is a translation of the French term *analyses des correspondances*, where the term *correspondance* denotes a "system of associations" between the elements of two sets. Originally developed by Jean-Paul Benzécri in the early 1960's, this exploratory technique has gained popularity in English-speaking countries in the late 1980's.

Generally speaking, correspondence analysis is a visualization technique used to represent the structure within matrices that contain some measure of correspondence between the rows and columns. This measure of correspondence is usually given as frequencies of items for a cross-classification of two categorical variables. Correspondence analysis is then used to create a plot where each row and column of the matrix is represented by one point, thus showing the interaction of two categorical variables. The space in which rows and columns are projected is usually one-, two-, or three-dimensional. In most cases, two-dimensional Euclidean space is used. Like in all other multivariate techniques that use SVD, axes spanning the space in which rows and columns are projected are not interpretable. If a row and a column point are close, that particular combination of categories of the two variables occurs more or less frequently than one would expect by chance, assuming that the two variables are independent.

One should note that as an exploratory technique, correspondence analysis is very similar to *factor analysis*, and some other multivariate statistical techniques like *principal component analysis, multi-dimensional scaling, reciprocal averaging*, etc. All these techniques are often used in text mining.

8.2 Applications of Correspondence Analysis

Correspondence analysis is a technique for displaying the rows and columns of a matrix as points in dual low-dimensional vector space. This way, the data can be displayed on a graph which is later used by human experts for further analysis.

In general, correspondence analysis can be used to perform the following types of analyses:

Discriminant analysis A given partition of i subjects into n groups is explored to find variables and patterns of observations that characterize and separate groups

Classification A set of grouped subjects is given from which classification of ungrouped observations should be inferred

Regression A dependency of one of the variables, called "dependent", is investigated regading the other "independent" variables. From the investigation, dependent variable can be forcasted for other values of independent variables than those observed

Cluster analysis Groups of similar objects are created by analyzing observations

For a detailed description on using correspondence analysis in the mentioned methods, one should refer to [22].

Although originally used to analyze textual data in linguistics, correspondence analysis has since been used in many other fields. For example, it was used by Krah et al. [31] in biology to study protein spots, while Zamir and Gabriel used [59] it to analyze time series on science doctorates in USA. Morin [36] used correspondence analysis for information retrieval on English abstracts of internal reports from a research center in France. She also explains why they prefer correspondence analysis to latent semantic indexing for this task. Some other applications of correspondence analysis include, for example, sociology [42]. A comprehensive list of publications prior to 1984 is given in [22].

8.3 Mathematical Background

In this section, mathematical foundations of correspondence analysis will be presented. First, some basic definitions from linear algebra that will be used thoroughout this chapter are given.

Definition 8.1. *A* field *is a set* F *together with two binary operations on* F*, called addition and multiplication, and denoted* $+$ *and* $\cdot$*, satisfying the following properties, for all* $a, b, c \in \mathrm{F}$*:*

1. $a+(b+c)=(a+b)+c$ *(associativity of addition)*
2. $a+b=b+a$ *(commutativity of addition)*
3. $a+0=a$ *for some element* $0 \in \mathrm{F}$ *(existence of zero element)*
4. $a+(-a)=0$ *for some element* $-a \in \mathrm{F}$ *(existence of additive inverses)*
5. $a\cdot(b\cdot c)=(a\cdot b)\cdot c$ *(associativity of multiplication)*
6. $a\cdot b=b\cdot a$ *(commutativity of multiplication)*
7. $a\cdot 1=a$ *for some element* $1 \in \mathrm{F}$*, with* $1 \neq 0$ *(existence of unit element)*
8. *If* $a \neq 0$*, then* $a\cdot a^{-1}=1$ *for some element* $a^{-1} \in \mathrm{F}$ *(existence of multiplicative inverses)*
9. $a\cdot(b+c)=(a\cdot b)+(a\cdot c)$ *(distributive property)*

Definition 8.2. *Let* $\mathbb{R}^2$ *be the set of all ordered pairs* (x, y) *of real numbers. We define two operations on* $\mathbb{R}^2$*, called addition and multiplication. They are defined as:*

$$+ : \mathbb{R}^2 \to \mathbb{R}^2$$

$$(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2) \tag{8.1}$$

$$\cdot : \mathbb{R}^2 \to \mathbb{R}^2$$

$$(x_1, y_1) \cdot (x_2, y_2) = (x_1 x_2 - y_1 y_2, x_1 y_2 + x_2 y_1) \tag{8.2}$$

Set $\mathbb{R}^2$ with operations $+$ and $\cdot$ is called field of complex numbers*, and is denoted by $\mathbb{C}$.*

It is easy to prove that operations $+$ and $\cdot$ satisfy the axioms of fields (1)–(9) in definition 8.1.

Definition 8.3. *A* matrix *is an* m*-by-*n *array of scalars from a field* F*. If* m=n*, the matrix is said to be square. The set of all* m*-by-*n *matrices over* F *is denoted by $M_{m,n}(\mathrm{F})$, and $M_{n,n}(\mathrm{F})$ is abbreviated to $M_n(\mathrm{F})$. In the most common case in which $\mathrm{F} = \mathbb{C}$, $M_n(\mathbb{C})$ is further abbreviated to M_n, and $M_{m,n}(\mathbb{C})$ to $M_{m,n}$. Elements of a matrix $\mathbf{M}$ will be denoted m_{ij}, where i denotes the row index, and j denotes the column index. Matrices will be denoted by capital letters.*

Definition 8.4. *Let $\mathbf{A}$ be an $m \times n$ matrix. The* conjugate transpose *of $\mathbf{A}$ is the $n \times m$ matrix defined as:*

$$\mathbf{A}^* \equiv \overline{\mathbf{A}}^{\mathrm{T}}, \tag{8.3}$$

where $\mathbf{A}^{\mathrm{T}}$ denotes the transpose of the matrix $\mathbf{A}$ and $\overline{\mathbf{A}}$ denotes the conjugate matrix (matrix obtained by taking the complex conjugate of each element of $\mathbf{A}$).

Definition 8.5. *Let $\mathbf{U}$ be an $n \times n$ complex matrix. $\mathbf{U}$ is said to be a* unitary matrix *if and only if it satisfies the following condition:*

$$\mathbf{U}^*\mathbf{U} = \mathbf{U}\mathbf{U}^* = \mathbf{I}_n, \tag{8.4}$$

where $\mathbf{U}^*$ denotes the conjugate transpose of $\mathbf{U}$.

Definition 8.6. *Let $\mathbf{M}$ be an $m \times n$ matrix. A non-negative real number σ is a* singular value *of $\mathbf{M}$ iff there exist unit-length vectors $\mathbf{u}$ and $\mathbf{v}$ such that*

$$\mathbf{M}\mathbf{v} = \sigma\mathbf{u} \quad \text{and} \quad \mathbf{M}^*\mathbf{u} = \sigma\mathbf{v}. \tag{8.5}$$

The vectors $\mathbf{u}$ and $\mathbf{v}$ are called left singular *and* right singular vectors *for σ, respectively.*

Theorem 8.1 (Singular value decomposition)**.** *Let* **M** *be an* $m \times n$ *matrix whose entries come from the field* K*, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization of the form*

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*, \tag{8.6}$$

i.e.,

$$\mathbf{M} = \sum_{k=1}^{p} \sigma_k \mathbf{u_k} \mathbf{v_k}^{\mathrm{T}} \tag{8.7}$$

where **U** *is an* $m \times m$ *unitary matrix over* K*,* $\boldsymbol{\Sigma}$ *is an* $m \times n$ *diagonal matrix of non-negative numbers, and* $\mathbf{V}^*$ *is the conjugate transpose of* **V***, an* $n \times n$ *unitary matrix over* K*. Such a factorization is called a* singular value decomposition *of* **M***. Note that the values on the diagonal of* $\boldsymbol{\Sigma}$ *(they will be denoted* σ_i*) are singular values of* **M***, and columns of* **U** *and* **V** *(*$\mathbf{u_i}$ *and* $\mathbf{v_i}$*, respectively) are left and right singular vectors for the corresponding singular values.*

Proof. The proof will be omitted as it is too long. It can be found for example in [57]. □

The SVD has one nice property. When the terms, corresponding to the smallest singular values, are dropped from the formula (8.7), a least-square approximation of the matrix **A** is obtained. That is, if we define the matrix $\mathbf{A_{[K]}}$ as the first K terms of (8.7):

$$\mathbf{A_{[K]}} \equiv \sum_{k=1}^{\mathrm{K}} \sigma_k \mathbf{u_k} \mathbf{v_k}^{\mathrm{T}} \tag{8.8}$$

then $\mathbf{A_{[K]}}$ minimizes

$$||\mathbf{A} - \mathbf{X}||^2 \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} - x_{ij})^2 \tag{8.9}$$

amongst all $m \times n$ matrices **X** of rank at most K. Note that $||\mathbf{A} - \mathbf{X}||^2$ is actually the squared Frobenius norm of the matrix $\mathbf{A}-\mathbf{X}$. $\mathbf{A_{[K]}}$ is called the rank K (least-squares) approximation of **A** and can itself be written in SVD form as:

$$\mathbf{A_{[K]}} = \mathbf{U_{(K)}} \boldsymbol{\Sigma}_{\mathbf{(K)}} \mathbf{V_{(K)}}^{\mathrm{T}} \tag{8.10}$$

where $\mathbf{U_{(K)}}$, $\mathbf{V_{(K)}}$, and $\boldsymbol{\Sigma}_{\mathbf{(K)}}$ are the relevant submatrices of **U**, **V**, and $\boldsymbol{\Sigma}$, respectively.

Matrix $\mathbf{A_{[K]}}$ is the least-squares approximation of **A** in the sense of Euclidean distance when the masses and dimension weights are absent. There is, however, a generalization of SVD that copes with masses and dimension weights.

Theorem 8.2 (Generalized SVD, GSVD)**.** *Let* $\boldsymbol{\Omega} \in \mathbb{R}^{m\times m}$ *and* $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$ *be positive-definite symmetric matrices. Then any real* $m \times n$ *matrix* $\mathbf{A}$ *of rank* K *can be expressed as*

$$\mathbf{A} = \mathbf{N}\mathbf{D}_\alpha \mathbf{M}^\mathrm{T} = \sum_{i=1}^{\mathrm{K}} \alpha_i \mathbf{n_i}\mathbf{m_i}^\mathrm{T} \tag{8.11}$$

where the columns of $\mathbf{N}$ *and* $\mathbf{M}$ *are orthonormalized with respect to* $\boldsymbol{\Omega}$ *and* $\boldsymbol{\Phi}$ *respectively:*

$$\mathbf{N}^\mathrm{T}\boldsymbol{\Omega}\mathbf{N} = \mathbf{M}^\mathrm{T}\boldsymbol{\Phi}\mathbf{M} = \mathbf{I}. \tag{8.12}$$

This decomposition is called generalized singular value decomposition "in the metrics" $\boldsymbol{\Omega}$ *and* Φ.

Proof. Let us observe the ordinary SVD of matrix $\boldsymbol{\Omega}^{1/2}\mathbf{A}\boldsymbol{\Phi}^{1/2}$:

$$\boldsymbol{\Omega}^{1/2}\mathbf{A}\boldsymbol{\Phi}^{1/2} = \mathbf{U}\mathbf{D}_\alpha\mathbf{V}^\mathrm{T}, \quad \text{where} \quad \mathbf{U}^\mathrm{T}\mathbf{U} = \mathbf{V}^\mathrm{T}\mathbf{V} = \mathbf{I}. \tag{8.13}$$

Letting $\mathbf{N} \equiv \boldsymbol{\Omega}^{-1/2}\mathbf{U}$ and $\mathbf{M} \equiv \boldsymbol{\Phi}^{-1/2}\mathbf{V}$ and substituting $\mathbf{U}$ and $\mathbf{V}$ in the previous equation, we get (8.11) and (8.12). □

If $\boldsymbol{\Omega} = \mathbf{D_w}$ is the matrix of masses, and $\boldsymbol{\Phi} = \mathbf{D_q}$ is the matrix of dimension weights, then the matrix approximation

$$\mathbf{A_{[K]}} = \mathbf{N_{(K)}}\mathbf{D}_{\mu\mathbf{(K)}}\mathbf{M_{(K)}}^\mathrm{T} = \sum_{k=1}^{\mathrm{K}} \mu_k \mathbf{n_k}\mathbf{m_k}^\mathrm{T} \tag{8.14}$$

minimizes

$$\begin{aligned} \|\mathbf{A}-\mathbf{X}\|^2_{\mathbf{D_q},\mathbf{D_w}} &\equiv \sum_{i=1}^{m}\sum_{j=1}^{n} w_i q_j (a_{ij} - x_{ij})^2 \\ &= \sum_{i=1}^{m} w_i (\mathbf{a_i} - \mathbf{x_i})^\mathrm{T}\mathbf{D_q}(\mathbf{a_i} - \mathbf{x_i}) \end{aligned} \tag{8.15}$$

amongst all matrices $\mathbf{X}$ of rank at most K. Note that μ_k in equation (8.14) denotes the elements of matrix $\mathbf{D}_\mu$ (that is, μ_k denotes the k-th singular value of $\mathbf{A_{[K]}}$). Further details about SVD and GSVD can be found in [22].

Computing the GSVD $\mathbf{A} = \mathbf{N}\mathbf{D}_\mu\mathbf{M}^\mathrm{T}$, where $\mathbf{N}^\mathrm{T}\mathbf{D_w}\mathbf{N} = \mathbf{M}^\mathrm{T}\mathbf{D_q}\mathbf{M} = \mathbf{I}$ is done in four steps:

1. Let $\mathbf{B} = \mathbf{D_w}^{1/2}\mathbf{A}\mathbf{D_q}^{1/2}$.
2. Find the ordinary SVD of $\mathbf{B}$: $\mathbf{B} = \mathbf{U}\mathbf{D}_\alpha\mathbf{V}^\mathrm{T}$.
3. Let $\mathbf{N} = \mathbf{D_w}^{-1/2}\mathbf{U}$, $\mathbf{M} = \mathbf{D_q}^{-1/2}\mathbf{V}$, $\mathbf{D}_\mu = \mathbf{D}_\alpha$.

4. Then $\mathbf{A} = \mathbf{N}\mathbf{D}_\mu\mathbf{M}^\mathrm{T}$ is the generalized SVD required.

We will now proceed to define some terms normally used in correspondence analysis.

Definition 8.7. *Let* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *be a matrix of non-negative numbers such that its row and column sums are non-zero, i.e.,*

$$\mathbf{A} \equiv [a_{ij}], a_{ij} \geq 0, \tag{8.16}$$

$$\sum_{i=1}^{m} a_{ij} > 0, \forall j \in \{1,\ldots,n\}, \tag{8.17}$$

$$\sum_{j=1}^{n} a_{ij} > 0, \forall i \in \{1,\ldots,m\}. \tag{8.18}$$

Let $a_{..}$ *denote the sum of all elements of* $\mathbf{A}$*:*

$$a_{..} \equiv \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}. \tag{8.19}$$

Correspondence matrix $\mathbf{P}$ *is defined as the matrix of elements of* $\mathbf{A}$ *divided by the grand total of* $\mathbf{A}$*:*

$$\mathbf{P} \equiv \frac{1}{a_{..}}\mathbf{A}. \tag{8.20}$$

The row and column sums of $\mathbf{P}$ *are* $m \times 1$ *and* $n \times 1$ *vectors* $\mathbf{r}$ *and* $\mathbf{c}$ *such that*

$$\mathbf{r} \equiv \mathbf{P}\mathbf{1}, r_i > 0, \forall i \in \{1,\ldots,m\} \tag{8.21}$$

and

$$\mathbf{c} \equiv \mathbf{P}^\mathrm{T}\mathbf{1}, c_j > 0, \forall j \in \{1,\ldots,n\}. \tag{8.22}$$

Diagonal matrices with elements on diagonal equal to elements of $\mathbf{r}$ *and* $\mathbf{c}$ *are denoted by* $\mathbf{D_r}$ *and* $\mathbf{D_c}$*, respectively:*

$$\mathbf{D_r} \equiv diag(\mathbf{r}) \quad \mathbf{D_c} \equiv diag(\mathbf{c}). \tag{8.23}$$

Note that $\mathbf{1} \equiv [1\ldots1]^\mathrm{T}$ denotes an $n \times 1$ or an $m \times 1$ vector, its order being deduced from particular context.

Definition 8.8. *Let* $\mathbf{P}$ *be a* $m \times n$ *correspondence matrix.* Row and column profiles *of* $\mathbf{P}$ *are vectors of rows and columns of* $\mathbf{P}$ *divided by their respective sums. Matrices of row and column profiles are denoted by* $\mathbf{R}$ *and* $\mathbf{C}$*, respectively:*

$$\mathbf{R} \equiv \mathbf{D_r}^{-1}\mathbf{P} = \begin{bmatrix} \tilde{\mathbf{r}}_1^\mathrm{T} \\ \vdots \\ \tilde{\mathbf{r}}_m^\mathrm{T} \end{bmatrix} \qquad \mathbf{C} \equiv \mathbf{D_c}^{-1}\mathbf{P} = \begin{bmatrix} \tilde{\mathbf{c}}_1^\mathrm{T} \\ \vdots \\ \tilde{\mathbf{c}}_n^\mathrm{T} \end{bmatrix} \tag{8.24}$$

The row and column profiles can be treated as points in respective n- and m-dimensional Euclidean spaces. Centroids of the row and column clouds in thier respective spaces are $\mathbf{c}$ and $\mathbf{r}$. Note that the dimension weights for the metric in these Euclidean spaces are defined by the inverses of the elements of $\mathbf{c}$ and $\mathbf{r}$, that is, $\mathbf{D}_c^{-1}$ and $\mathbf{D}_r^{-1}$, respectively.

Definition 8.9. *Let* $\mathbf{R}$ *and* $\mathbf{C}$ *be the matrices of row and column profiles of a correspondence matrix* $\mathbf{P}$*, thereby defining two clouds of points in their respective* n*- and* m*- dimensional space. For each cloud, the weighted sum of squared distances from points to thier respective centroids is called* total inertia *of that cloud. Total inertia for row points is*

$$inertia(\mathbf{R}) = \sum_{i=1}^{m} r_i(\tilde{\mathbf{r}}_i - \mathbf{c})^{\mathrm{T}}\mathbf{D}_c^{-1}(\tilde{\mathbf{r}}_i - \mathbf{c}) \tag{8.25}$$

$$inertia(\mathbf{R}) = trace[\mathbf{D}_r(\mathbf{R} - \mathbf{1c}^{\mathrm{T}})\mathbf{D}_c^{-1}(\mathbf{R} - \mathbf{1c}^{\mathrm{T}})^{\mathrm{T}}], \tag{8.26}$$

while for column points it is

$$inertia(\mathbf{C}) = \sum_{j=1}^{n} c_j(\tilde{\mathbf{c}}_j - \mathbf{r})^{\mathrm{T}}\mathbf{D}_r^{-1}(\tilde{\mathbf{c}}_j - \mathbf{r}) \tag{8.27}$$

$$inertia(\mathbf{C}) = trace[\mathbf{D}_c(\mathbf{C} - \mathbf{1r}^{\mathrm{T}})\mathbf{D}_r^{-1}(\mathbf{C} - \mathbf{1r}^{\mathrm{T}})^{\mathrm{T}}]. \tag{8.28}$$

The notation *trace* is used to denote the matrix trace operator. Note that the term *inertia* in correspondence analysis is used by analogy with the definition in applied mathematics of *moment of inertia*, which stands for the integal of mass times the squared distance to the centroid.

Theorem 8.3. *Let* $in(\mathbf{R})$ *and* $in(\mathbf{C})$ *be the total inertia of row and column cloud, respectively. When calculated, the total inertia is the same for both clouds, and it is equal to the mean-square contingency coefficient calculated on the original matrix* $\mathbf{A}$.

$$\begin{aligned} in(\mathbf{R}) = in(\mathbf{C}) &= \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{(p_{ij} - r_i c_j)^2}{r_i c_j} = \frac{\chi^2}{a_{..}} \\ &= \mathrm{trace}[\mathbf{D_r}^{-1}(\mathbf{P} - \mathbf{rc}^{\mathrm{T}})\mathbf{D_c}^{-1}(\mathbf{P} - \mathbf{rc}^{\mathrm{T}})^{\mathrm{T}}] \end{aligned} \tag{8.29}$$

$$\chi^2 \equiv \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{(a_{ij} - e_{ij})^2}{e_{ij}} \tag{8.30}$$

$$e_{ij} \equiv \frac{a_{i.}a_{.j}}{a_{..}} \tag{8.31}$$

Proof. From (8.25) and (8.27) we have

$$in(\mathbf{R}) = \sum_{i=1}^{m} r_i \sum_{j=1}^{n} (p_{ij}/r_i - c_j)^2/c_j = \sum_{i=1}^{m}\sum_{j=1}^{n} (p_{ij} - r_i c_j)^2/r_i c_j$$

and

$$in(\mathbf{C}) = \sum_{j=1}^{n} c_j \sum_{i=1}^{m} (p_{ij}/c_j - r_i)^2/r_i = \sum_{j=1}^{n}\sum_{i=1}^{m} (p_{ij} - r_i c_j)^2/r_i c_j.$$

Hence

$$in(\mathbf{R}) = in(\mathbf{C}).$$

In the χ^2 formula $a_{ij} = a_{..} p_{ij}$ and thus the "expected" value in a cell is:

$$e_{ij} \equiv (\sum_{j=1}^{n} a_{ij})(\sum_{i=1}^{m} a_{ij})/a_{..} = (a_{..} r_i)(a_{..} c_j)/a_{..} = a_{..} r_i c_j.$$

This implies that $\chi^2 = a_{..} in(\mathbf{R}) = a_{..} in(\mathbf{C})$, hence (8.29). □

The respective K-dimensional subspaces of row and column clouds which are the closest to the points in terms of weighted sum of squared distances are defined by K right and left (generalized) singular vectors of $\mathbf{P} - \mathbf{r}\mathbf{c}^{\mathrm{T}}$ in metrics $\mathbf{D_c}^{-1}$ and $\mathbf{D_r}^{-1}$ which correspond to the K largest singular values.

Let the generalized SVD of $\mathbf{P} - \mathbf{r}\mathbf{c}^{\mathrm{T}}$ be

$$\mathbf{P} - \mathbf{r}\mathbf{c}^{\mathrm{T}} = \mathbf{A}\mathbf{D}_{\mu}\mathbf{B}^{\mathrm{T}}, \quad \text{where} \quad \mathbf{A}^{\mathrm{T}}\mathbf{D_r}^{-1}\mathbf{A} = \mathbf{B}^{\mathrm{T}}\mathbf{D_c}^{-1}\mathbf{B} = \mathbf{I} \quad \text{and} \quad \mu_1 \geq \ldots \geq \mu_k > 0 \tag{8.32}$$

Then the columns of **A** and **B** define the principal axes of the row and column clouds, respectively.

Theorem 8.4. *Let* **R** *and* **C** *be the matrices of row and column profiles. Then the coordinates of vectors from* **R** *and* **C** *with respect to their own principal axes are related to the principal axes of the other cloud of profiles by simple rescaling. Let*

$$\mathbf{F} \equiv (\mathbf{R} - \mathbf{1}\mathbf{c}^{\mathrm{T}})\mathbf{D_c}^{-1}\mathbf{B} = (\mathbf{D_r}^{-1}\mathbf{P} - \mathbf{1}\mathbf{c}^{\mathrm{T}})\mathbf{D_c}^{-1}\mathbf{B} \tag{8.33}$$

be the coordinates of row profiles with respect to principal axes **B** *in the* χ^2 *metric* $\mathbf{D_c}^{-1}$. *Then*

$$\mathbf{F} = \mathbf{D_r}^{-1}\mathbf{A}\mathbf{D}_{\mu}. \tag{8.34}$$

Let

$$\mathbf{G} \equiv (\mathbf{C} - \mathbf{1}\mathbf{r}^{\mathrm{T}})\mathbf{D_r}^{-1}\mathbf{A} = (\mathbf{D_c}^{-1}\mathbf{P}^{\mathrm{T}} - \mathbf{1}\mathbf{r}^{\mathrm{T}})\mathbf{D_r}^{-1}\mathbf{A} \tag{8.35}$$

be the coordinates of the column profiles with respect to principal axes **A** *in the* χ^2 *metric* $\mathbf{D_r}^{-1}$. *Then*

$$\mathbf{G} = \mathbf{D_c}^{-1}\mathbf{B}\mathbf{D}_{\mu}. \tag{8.36}$$

Proof. Let us consider the coordinates of row profiles. Notice that, since principal axes $\mathbf{B}$ are orthonormal (see (8.32)), these coordinates are just scalar products of the centered profiles with $\mathbf{B}$, hence the definition in (8.33). Using $\mathbf{1} = \mathbf{D_r}^{-1}\mathbf{r}$ we can rewrite (8.33) as follows:

$$\mathbf{F} = \mathbf{D_r}^{-1}(\mathbf{P} - \mathbf{rc}^{\mathrm{T}})\mathbf{D_c}^{-1}\mathbf{B}. \tag{8.37}$$

Multiplying the generalized SVD of $\mathbf{P} - \mathbf{rc}^{\mathrm{T}}$ on the right by $\mathbf{D_c}^{-1}\mathbf{B}$ we obtain:

$$(\mathbf{P} - \mathbf{rc}^{\mathrm{T}})\mathbf{D_c}^{-1}\mathbf{B} = \mathbf{AD}_{\mu}. \tag{8.38}$$

Substituting $(\mathbf{P} - \mathbf{rc}^{\mathrm{T}})\mathbf{D_c}^{-1}\mathbf{B}$ in (8.37) with $\mathbf{AD}_{\mu}$ we get (8.34). □

As an immediate consequence of the previous theorem and (8.32), the two sets of coordinates ($\mathbf{F}$ and $\mathbf{G}$) are related to each other by the following formulæ:

$$\mathbf{G} = \mathbf{D_c}^{-1}\mathbf{P}^{\mathrm{T}}\mathbf{F}\mathbf{D}_{\mu}^{-1} = \mathbf{CFD}_{\mu}^{-1}, \quad \text{i.e.,} \quad \mathbf{GD}_{\mu} = \mathbf{D_c}^{-1}\mathbf{P}^{\mathrm{T}}\mathbf{F} \tag{8.39}$$

$$\mathbf{F} = \mathbf{D_r}^{-1}\mathbf{PGD}_{\mu}^{-1} = \mathbf{RGD}_{\mu}^{-1}, \quad \text{i.e.,} \quad \mathbf{FD}_{\mu} = \mathbf{D_r}^{-1}\mathbf{PG} \tag{8.40}$$

Theorem 8.5. *With respect to the principal axes, the respective clouds of row and column profiles have centroids at the origin. The weighted sum of squares of the points' coordinates along the kth principal axis in each cloud is equal to μ_k^2, which is denoted by λ_k and called the kth* principal inertia. *The weighted sum of cross-products of the coordinates is zero.*

Centroid of rows of $\mathbf{F}$*:* *Principal inertias of row cloud:*

$$\mathbf{r}^{\mathrm{T}}\mathbf{F} = \mathbf{0}^{\mathrm{T}} \qquad \mathbf{F}^{\mathrm{T}}\mathbf{D_r}\mathbf{F} = \mathbf{D}_{\mu}{}^{2} \equiv \mathbf{D}_{\lambda} \tag{8.41}$$

Centroid of rows of $\mathbf{G}$*:* *Principal inertias of column cloud:*

$$\mathbf{c}^{\mathrm{T}}\mathbf{G} = \mathbf{0}^{\mathrm{T}} \qquad \mathbf{G}^{\mathrm{T}}\mathbf{D_c}\mathbf{G} = \mathbf{D}_{\mu}{}^{2} \equiv \mathbf{D}_{\lambda} \tag{8.42}$$

Proof. The proof will be omitted as it is trivial. □

The total inertia of each cloud of points is decomposed along the principal axes and amongst the points themselves in a similar and symmetric fashion. This gives a decomposition of inertia for each cloud of points which is analogous to a decomposition of variance. This decomposition is shown in table 8.1 on the next page.

This table forms the numerical support for the graphical display. The columns display contributions of the rows and columns to the inertia of an axis. Each of these contributions can be expressed as a proportion of the respective inertia λ_k ($\equiv \mu_k^2$) in order to interpret the axis. These contributions are called

TABLE 8.1: Decomposition of inertia

		1	2	…	K	total
	1	$r_1 f_{11}^2$	$r_1 f_{12}^2$	…	$r_1 f_{1\mathrm{K}}^2$	$r_1 \sum_k f_{1\mathrm{K}}^2$
	2	$r_2 f_{21}^2$	$r_2 f_{22}^2$	…	$r_2 f_{2\mathrm{K}}^2$	$r_2 \sum_k f_{2\mathrm{K}}^2$
rows	⋮	⋮	⋮	…	⋮	⋮
	m	$r_m f_{m1}^2$	$r_m f_{m2}^2$	…	$r_m f_{m\mathrm{K}}^2$	$r_m \sum_k f_{m\mathrm{K}}^2$
	total	$\lambda_1 \equiv \mu_1^2$	$\lambda_2 \equiv \mu_2^2$	…	$\lambda_\mathrm{K} \equiv \mu_\mathrm{K}^2$	$in(\mathbf{R}) = in(\mathbf{C})$
	1	$c_1 g_{11}^2$	$c_1 g_{12}^2$	…	$c_1 g_{1\mathrm{K}}^2$	$c_1 \sum_k g_{1\mathrm{K}}^2$
	2	$c_2 g_{21}^2$	$c_2 g_{22}^2$	…	$c_2 g_{2\mathrm{K}}^2$	$c_2 \sum_k g_{2\mathrm{K}}^2$
columns	⋮	⋮	⋮	…	⋮	⋮
	n	$c_n g_{n1}^2$	$c_n g_{n2}^2$	…	$c_n g_{n\mathrm{K}}^2$	$c_n \sum_k g_{n\mathrm{K}}^2$

"absolute" because they are affected by the mass of each point. Each row of the table contains the contributions of the axes to the inertia of the respective profile point. And again, these contributions express proportions of the point's inertia in order to interpret how well the point is represented on the axes. These are called "relative contributions" because the masses are devided out.

It is interesting to note here that the correspondence matrix **P** can be reconstituted from the matrices **F**, **G**, and $\mathbf{D}_\mu$ using the formula (8.43), while a K rank approximation of **P** can be computed from (8.44).

$$\mathbf{P} = \mathbf{rc}^\mathrm{T} + \mathbf{D_r F D}_\mu{}^{-1}\mathbf{G}^\mathrm{T}\mathbf{D_c} \tag{8.43}$$

$$\approx \mathbf{rc}^\mathrm{T} + \mathbf{D_r F}_{(\mathrm{K})}\mathbf{D}_{\mu(\mathrm{K})}^{-1}\mathbf{G}_{(\mathrm{K})}^\mathrm{T}\mathbf{D_c} \tag{8.44}$$

To illustrate the mathematical concepts introduced in this section, consider the following example:

Example 8.1. *Correspondence analysis of smoker data*
Suppose we have data on the smoking habits of different employees in a company. The data is given in table 8.2.

The correspondence matrix **P** is then

$$\mathbf{P} = \begin{bmatrix} 0.0207 & 0.0104 & 0.0155 & 0.0104 \\ 0.0207 & 0.0155 & 0.0363 & 0.0207 \\ 0.1295 & 0.0518 & 0.0622 & 0.0207 \\ 0.0933 & 0.1244 & 0.1710 & 0.0674 \\ 0.0518 & 0.0311 & 0.0363 & 0.0104 \end{bmatrix}$$

while the row and column sums are

TABLE 8.2: Incidence of smoking amongst five different types of staff. The data is taken from [22]

	Smoking Category				
Staff Group	None	Light	Medium	Heavy	Total
Senior Managers	4	2	3	2	11
Junior Managers	4	3	7	4	18
Senior Employees	25	10	12	4	51
Junior Employees	18	24	33	13	88
Secretaries	10	6	7	2	25
Total	61	45	62	25	193

$$\mathbf{r} = \begin{bmatrix} 0.3161 & 0.2332 & 0.3212 & 0.1295 \end{bmatrix}^{\mathrm{T}}$$

and

$$\mathbf{c} = \begin{bmatrix} 0.0570 & 0.0933 & 0.2642 & 0.4560 & 0.1295 \end{bmatrix}.$$

The row and column profiles are vectors of rows and columns of **P** divided by their respective sums (elements of **r** and **c**):

$$\mathbf{R} = \begin{bmatrix} 0.3636 & 0.1818 & 0.2727 & 0.1818 \\ 0.2222 & 0.1667 & 0.3889 & 0.2222 \\ 0.4902 & 0.1961 & 0.2353 & 0.0784 \\ 0.2045 & 0.2727 & 0.3750 & 0.1477 \\ 0.4000 & 0.2400 & 0.2800 & 0.0800 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0.6565 & 0.6565 & 0.4098 & 0.2950 & 0.1639 \\ 0.0444 & 0.0667 & 0.2222 & 0.5332 & 0.1333 \\ 0.0484 & 0.1129 & 0.1936 & 0.5323 & 0.1129 \\ 0.0800 & 0.1600 & 0.1600 & 0.5200 & 0.0800 \end{bmatrix}.$$

Row profiles indicate what smoking patterns different types of employees follow (analogous for column points). For example, it is obvious from **R** that *Senior employees* and *Secretaries* exhibit very similar patterns of relative frequencies across the categories of smoking intensity.

The inertia is same for the row and column cloud and it is equal to 0.08519. Note that the total value of χ^2 is 16.442 (i.e., 0.08519*193, according to equation (8.29)). After computing generalized SVD of $\mathbf{P} - \mathbf{r}\mathbf{c}^{\mathrm{T}}$ we can compute the coordinates for row and column points in the new space. The coordinates for the first two dimensions are (**F** and **G** are coordinates of row and column points, respectively):

$$\mathbf{F} = \begin{bmatrix} -0.065768 & 0.193737 \\ 0.258958 & 0.243305 \\ 0.380595 & 0.010660 \\ 0.232952 & -0.057744 \\ -0.201089 & -0.078911 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} -0.393308 & 0.030492 \\ 0.099456 & -0.141064 \\ 0.196321 & -0.007359 \\ 0.293776 & 0.197766 \end{bmatrix}.$$

These first two dimensions (corresponding to two largest singular values of $\mathbf{P} - \mathbf{rc}^{\mathrm{T}}$) explain 99.5145% of inertia. This means that the relative frequency value that can be reconstructed from these two dimensions can reproduce 99.5145% of the total χ^2 value for this two-way table. The biplot of the smokers data in the first two dimensions is shown in figure 8.1.

From figure 8.1 one can see that the categoy *None* is the only column point on the left side of the origin for the first axis. Since emplyee group *Senior Employees* also falls on that side of the first axis, one may conclude that the first axis separates *None* smokers from the other categories of smokers, and that *Senior Employees* are different from, e.g., *Junior Employees*, in that there are relatively more non-smoking *Senior Employees.* Also, the proximity of *Heavy* smoker catrgory and *Junior Managers* employee type suggest a larger portion of heavy smokers amongst junior managers than one would normally expect.

Example 8.1 not only demonstrates the mathematical concepts of correspondence analysis, but also shows how this exploratory technique is used, i.e., how the biplot is interpreted and what possible conclusions can be drawn from it.

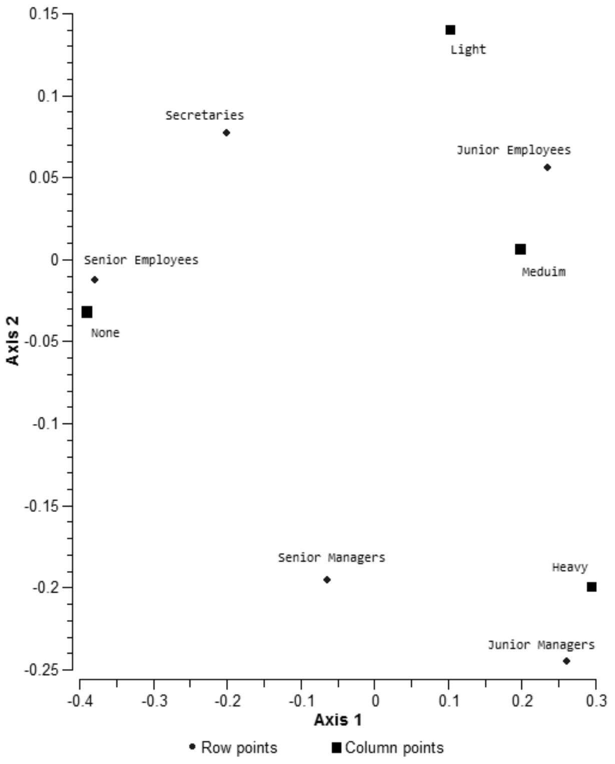


Figure 8.1: Biplot showing employee types and smoker categories

CHAPTER 9

Implementation in Orange

C++ is an insult to the human brain

Niklaus Wirth

Orange [13] is a component-based open-source data mining software developed at AI Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Slovenia. It includes a number of preprocessing, modelling and data exploration techniques. Orange can be downloaded from `http://www.ailab.si/orange/`. Orange is written mostly in Python,* except for its kernel which is written in C++.

This chapter will describe the implementation of text preprocessing methods in Orange, and will also give useful examples on using them. First, in sections 9.1 and 9.2, functions for text preprocessing and feature selection that were written on the script level of Orange will be described. As the real advantage of using Orange is the simplicity of using its widgets for visual programming, widgets were created that provide the functionality described in the first two sections. These widgets are covered in section 9.3.

9.1 Text Preprocessing in Orange

All the classes and functions for working with text in Orange are united under the module **orngText**. This section will describe those functions and show how they can be used from within scripts or the Python interactive shell.

*`http://www.python.org/`

Listing 9.1: Structure of DocumentSet XML

```
<set name = "...">
  <document id = "...">
    <categories>
      <category> ... </category>
       ...
      <category> ... </category>
    </categories>
    <content>
     ...
    </content>
     ...
  </document>
   ...
</set>
```

9.1.1 Loading Textual Data

Module **orngText** accepts various textual formats as input: XML, pure text, and Reuters .sgm files. Following is an overview of the functions for loading these formats.

loadFromListWithCategories(fileName)

This function will load pure textual data from file **fileName**. **fileName** is a file that has two lines for each document to be loaded—first line contains the path to the document and the second line contains space separated categories. If a document's category is unknown, the second line for that document has to be left blank.

loadFromXML(fileName, tags={}, doNotParse=[])

Loads textual data from XML file **fileName** into an ExampleTable and returns that table. XML should be of DocumentSet type. The structure of this XML is shown in listing 9.1. The tag **set** is the top level tag of the XML, which contains a collection of documents. Each document is enclosed within **document** tag. Each document may (but doesn't have to) have categories associated with it. The categories are placed inside the **categories** tag. **categories** tag can hold one or more **category** tag, each of them containing one category for the document. The content of the document (its text) is placed within **content** tag. Other tags can be placed within **content** tag or after it but they will be ignored unless stated otherwise.

If provided, the dictionary **tags** changes the standard names of tags. For example, to change the tag for the begining of a new document from 'document' to 'doc', and leave the other tags intact, simply put tags = {'document': 'doc'}. Tags provided in the list **doNotParse** will not be parsed. If omitted, every tag will be parsed.

loadReuters(dirName)

Loads all .sgm files from directory **dirName** into an ExampleTable and returns it. Sgm files have an XML-like structure and support for them is included because the Reuters 21578* collection comes in that format.

After loading the text into Orange data structures, some basic preprocessing can be done with it. Following is a list of functions that make this preprocessing possible.

9.1.2 Preprocessing the Text

Before adding textual features, the text has to be preprocessed in some way. This preprocessing includes tokenization, stopword removal, and lemmatization, and it is done using the class **Preprocess**, which will now be described.

class Preprocess(object)

Class for constructing objects that serve for preprocessing of text (lemmatization, stopwords removing, and tokenization). Note that this class does not add any features, it changes the text in the original ExampleTable.

Attributes

inputEncoding

String indicating the input encoding of the text. For a list of all possible values, see [1, pp. 208–210].

outputEncoding

String indicating the output encoding of the text. For a list of all possible values, see [1, pp. 208–210].

lemmatizer

Function used to perform the lemmatization of text. It should take a string as an argument and return a lemmatized string.

stopwords

A set of stopwords, i.e., words that will be removed during preprocessing.

tokenize

Function used to tokenize the words in the text. It should take a string (text) as an argument and return a list of words from that text.

*This is a collection widely used in many text mining applications as a standard data set for experiments. It can be found at http://www.daviddlewis.com/resources/testcollections/reuters21578/

langData
: A dictionary of predefined lemmatizers, tokenizers, and lists of stopwords for each of the currently supported languages (Croatian, English, and French). All three languages share the same tokenizer provided by the TMT [47] library. TMT (*Text Mining Tools*) is a C++ library of classes and routines for preprocessing corpora of textual documents. There are also some machine learning algorithms implemented in TMT. TMT is developed at Faculty of Electrical Engineering and Computing, University of Zagreb, by students Frane Šarić and Artur Šilić, under supervision of prof. Bojana Dalbelo Bašić and Jan Šnajder. Lemmatizer for Croatian is based on the automatically generated morphological dictionary (see [48]). For English, Porter's algorithm [43] was used. Both of these lemmatizers are integrated in the TMT library. For use in Python, a wrapper was generated using SWIG.* Unfortunately, there is no publicly available lemmatizer for French, known to the author, so a *NOP lemmatizer*† was used. Lists of stopwords are specific for each language.

Methods

__init__(self, language=None, inputEncoding='cp1250', outputEncoding='cp1250')
: Constructor for Preprocess class. **language** can be either 'en', 'hr', or 'fr'. **inputEncoding** and **outputEncoding** are strings, for a list all possible values, see [1, pp. 208–210].

_in2utf(self, s)
: Converts string **s** from **inputEncoding** to UTF-8.

_utf2out(self, s)
: Converts string **s** from UTF-8 to **outputEncoding**.

doOnExampleTable(self, data, textAttributePos, meth)
: Executes function **meth** for each example in ExampleTable **data**. **meth** is executed on the attribute specified by **textAttributePos** argument.

lemmatizeExampleTable(self, data, textAttributePos)
: Lemmatizes each example in ExampleTable **data**. The text attribute is specified by the **textAttributePos** argument.

lemmatize(self, token)
: Lemmatizes a single token or a list of tokens.

removeStopwordsFromExampleTable(self, data, textAttributePos)
: Removes the stopwords from each example in ExampleTable **data**. The text attribute is specified by the **textAttributePos** argument.

removeStopwords(self, token)

*http://www.swig.org/

†A lemmtizer that leaves each word unchanged.

Removes stopwords from text or a list of tokens.

9.1.3 Adding Textual Features

Following is a description of functions that enable adding of different features (words, letter n-grams, and word n-grams) to textual data in Orange.

bagOfWords(exampleTable, preprocessor=None, stopwords=None)

Adds words as features in ExampleTable **exampleTable**. **preprocessor** is an instance of a Preprocess class and it will, if provided, preprocess the text in the desired way before constructing word features (see documentation for Preprocess class earlier in this section for more). **stopwords** is a Python **set** object containing words that should not be added as features (i.e., stopwords).* For Python versions earlier than 2.5, **Sets.set** should be used instead of **set**.

Example 9.1. *Adding words as features*

Suppose we have an ExampleTable **data** for which we wish to add words as features. No special text preprocessing will be done, other than tokenization. The following code demonstrates how this is done.

Listing 9.2: Adding words as features

```
>>> data
<ExampleTable instance at 0x00C9AE68>
>>> data[0]
['', '', '', 'Mary had a little lamb.']
>>> res = orngText.bagOfWords(data)
>>> res[0]
['', '', '', 'Mary had a little lamb.'], {"Mary":1.000, "had":
1.000, "a":1.000, "little":1.000, "lamb":1.000}
```

extractLetterNGram(table, n=2)

Adds letter n-grams as features to ExampleTable **table**. **n** is the length of n-grams to be added. All characters are treated equally, i.e., punctuations, digits, and other non-alphabet characters are included in letter n-grams.

Example 9.2. *Adding letter n-grams as features for text*

Suppose we have an ExampleTable **data** for which we wish to add letter n-grams as features. The following code demonstrates how this is done.

*A careful reader will notice that the functionality provided by **stopwords** argument can also be achieved through the **preprocessor** argument, provided the options are set accordingly. However, specifying that we only wish some stopwords removed from the list of words is more easier done through providing those words in a list as an argument rather than constructing a **Preprocess** object and setting the appropriate options. Don't try to kill a fly with a hand grenade.

Listing 9.3: Adding letter n-grams

```
>>> data
<ExampleTable instance at 0x00C9AE68>
>>> data[0]
['', '', '', 'Mary had a little lamb.']
>>> res = orngText.extractLetterNGram(data, 2)
>>> res[0]
['', '', '', 'Mary had a little lamb.'], {"a ":1.000, "e ":1.000,
" a":1. 000, "Ma":1.000, "ad":1.000, "la":1.000, "y ":1.000,
" h":1.000, "mb":1.000, "am ":1.000, "ry":1.000, "d ":1.000,
"li":1.000, "le":1.000, "tl":1.000, "ar":1.000, " l":2.000,
"it":1.000, "ha":1.000, "tt":1.000, "b.":1.000}
```

extractWordNGram(table, preprocessor = None, n = 2, stopwords = None, threshold = 4, measure = 'FREQ')

Add word n-grams as features to ExampleTable **table**. If provided, **preprocessor** will preprocess the text in the desired manner before adding the features. **n** is the number of words in the n-gram, and it defaults to two. Set of words provided as **stopwords** will greatly improve the quality of word n-grams, but this argument is optional. All n-grams having value of the given association measure above **threshold** will be kept as features, while others will be discarded. **measure** is a function that indicates how strongly the words in the n-gram are associated.* The higher this value, the stronger the association. **measure** can have the following values: 'FREQ', 'CHI', 'DICE', 'LL', and 'MI'. 'FREQ' will assign each n-gram its frequency in the data. 'CHI', 'DICE', 'LL', and 'MI' will compute for each n-gram its chi-squared value, Dice coefficient, log-likelihood value, and mutual information, respectively. These measures are described in more detail in chapter 3.

9.2 Feature Selection

Simply adding text features is not enough for any serious application. This is because there are normally tens of thousands of features (or more) and the time it would take to process them all is just not feasible. Not only that, but the results obtained by using all the features in some application would not be very good as many of the features simply aren't representative for the text. Using all the features of a text to, for example, predict its category would be the same as using all the information we have about a person (including his eye

*That is, an association measure.

color, shirts he wore yesterday, name of his uncle, etc.) to predict whether or not he would make a good employee in a software firm. Therefore, selecting only a subset of features is always an important task in text mining. In this section functions developed for this purpose will be described.

FSS(table, funcName, operator, threshold, perc = True)

Removes text features from **table**, using function **funcName**, operator **operator**, and the threshold **threshold**. If **perc** is **True**, then the number **threshold** is the percentage of features to be removed, otherwise it is regarded as a threshold and all features having value of **funcName** below (or above) threshold are removed. **funcName** can be one of the following: 'TF', 'TDF', and 'RAND'. 'TF'(term frequency) is a function that returns the number of times a feature (term) appears in the data. 'TDF'(term document frequency) is a function that returns the number of documents a feature appears in, while 'RAND' gives a random value to each feature. **operator** can be 'MIN' or 'MAX'. With 'MIN' specified as the operator, this function will remove the features with value of **funcName** less than **threshold** (or the **threshold** percent of features with the lowest values, in case **perc** is **True**). Specifying 'MAX' will do the opposite. For example, keeping only the most frequent 10% of features is done with

```
res = orngText.FSS(data, 'TF', 'MIN', 0.9, True)
```

Removing the features that appear in more than 50 documents is done with

```
res = orngText.FSS(data, 'TDF', 'MAX', 50, False)
```

DSS(table, funcName, operator, threshold)

Function for document subset selection. Takes an ExampleTable **table**, function **funcName**, operator **operator**, and a number **threshold** and removes all the documents that have the value of **funcName** below (or above) **threshold**. **funcName** can be 'WF' or 'NF'. 'WF'(word frequency) is a function that returns the number of words a document has. 'NF'(number of features) is a function that returns the number of different features a document has. For example, if a document would consist only of the sentence "Mary also loves her mother, which is also named Mary.", its 'WF' would be 10, and its 'NF' would be 8. **operator** can be either 'MIN' or 'MAX'. If 'MIN' is chosen, the function will remove all documents having the value of **funcName** less than **threshold**. 'MAX' behaves the opposite. Removing all documents that have less than 10 features is done with

```
res = orngText.DSS(data, 'WF', 'MIN', 10)
```

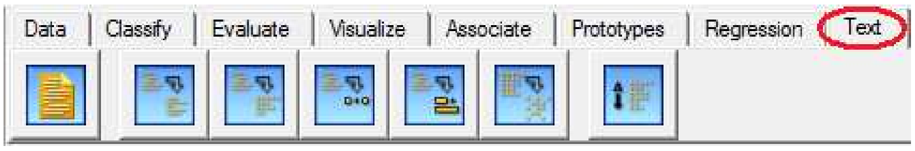
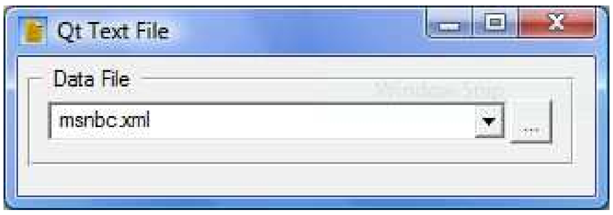


Figure 9.1: Text tab in Orange containing widgets for text preprocessing



msnbc.xml

Figure 9.2: TextFile widget

9.3 Visual Programming with Widgets

Orange widgets are used for visual programming. Each wigdet has its input and output channels used for communication with other widgets. Programming with wigdets is done in Orange Canvas by connecting widgets and setting each widgets' properties. Details about Orange widgets can be found in Orange's documentation.* For each function described in the previous section, there is a widget that incorporates its functionality. Those widgets will be described here. Figure 9.1 shows the Text tab in Orange where all the widgets for manipulating text are found.

9.3.1 TextFile widget

Inputs

(none)

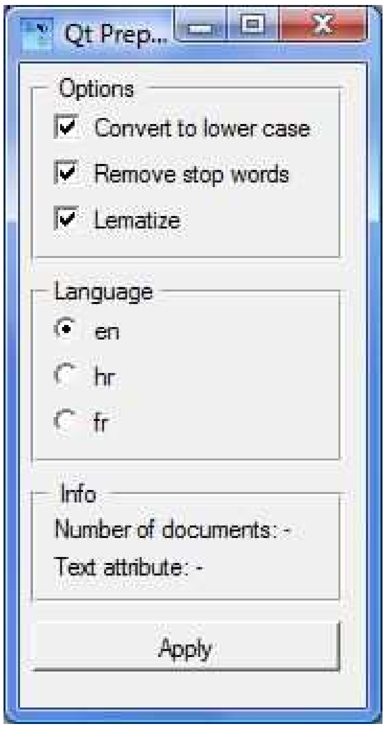
Outputs

Documents (ExampleTable)

Description

This widget is used to load textual data into an ExampleTable. It accepts data in XML format, pure text, and .sgm format. Widget is displayed in figure 9.2.

*http://www.ailab.si/orange/doc/widgets/



☑ Convert to lower case
☑ Remove stop words
☑ Lematize

Language
(●) en
( ) hr
( ) fr

Info
Number of documents: -
Text attribute: -

Figure 9.3: Preprocess widget

9.3.2 TextPreprocess widget

Inputs
Examples (ExampleTable)
Outputs
Examples (ExampleTable)
Description
Constructs an **orngText.Preprocess** object and uses it to process the text in the desired way. Widget is displayed in figure 9.3. User can choose whether or not to convert words to lower case, remove stopwords, and lemmatize. These three options are available for English and Croatian. For French, lemmatizing is not available because at the time of writing no French morphological dictionary was available to the author.

9.3.3 BagOfWords widget

Inputs
Examples (ExampleTable)
Outputs

Examples (ExampleTable)

Description

Constructs the standard bag-of-words representation of a text, i.e., it adds words as features that represent text. There are some options available through this widget. Choosing the “log(1/f)” option in the TFIDF box computes the TFIDF* of a feature and uses that value to represent a document instead of the features’ frequency which is used by default. The TFIDF is a statistical measure, often used in text mining, that evaluates how important a term is to a document in a corpus. Importance of a term for a specific document is proportional to the number of times the word appears in the document, but is offset by the frequency of the term in the corpus. TFIDF is computed as:

$$\text{tfidf} = \underbrace{\frac{n_i}{\sum_k n_k}}_{\text{tf}} \underbrace{\log \frac{|\text{D}|}{|\{d : t_i \ni d\}|}}_{\text{idf}}, \tag{9.1}$$

where n_i is the number of occurrences of the considered term, $\sum_k n_k$ is the number of occurrences of all terms, |D| is the total number of documents in the corpus, and $|\{d : t_i \ni d\}|$ is the number of documents where the term t_i appears.

It is also possible to normalize the length of a document. Normalizing the length of a document means that the vector representing this document in feature space will have the chosen norm equal to one. Currently two different norms are accepted: L1 (Manhattan) and L2 (Euclidean). L1 norm of a vector $\mathbf{v} \in \mathbb{R}^n$ is defined as:

$$\text{L1}(\mathbf{v}) = \sum_{i=1}^{n} v_i, \tag{9.2}$$

while L2 is defined as:

$$\text{L2}(\mathbf{v}) = \sqrt{\sum_{i=1}^{n} v_i^2}. \tag{9.3}$$

The bottom of the widget shows some basic information about the data. Widget is shown in figure 9.4.

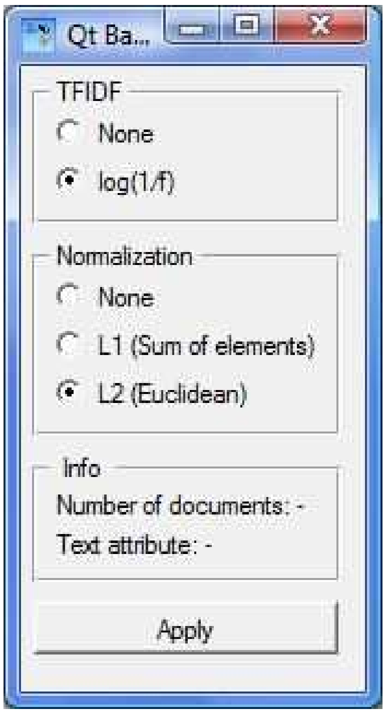
9.3.4 LetterNgram widget

Inputs

Examples (ExampleTable)

Outputs

*Term frequency–inverse document frequency



TFIDF
- ( ) None
- (•) log(1/f)

Normalization
- ( ) None
- ( ) L1 (Sum of elements)
- (•) L2 (Euclidean)

Info
Number of documents: -
Text attribute: -

Figure 9.4: Bag of words widget

Examples (ExampleTable)

Description

Constructs letter n-grams as features for text. Letter n-grams of two, three, and four letters can be chosen. Bottom of the widget shows the number of different letter n-grams found in the data. Widget is shown in figure 9.5.

9.3.5 WordNgram widget

Inputs

Examples (ExampleTable)

Outputs

Examples (ExampleTable)

Description

Constructs the word n-gram representation of text. Some word n-grams—collocations—are especially interesting. Collocations and their extraction from text have been the topic of the first part of this thesis. One can choose to extract word n-grams of two, three, and four words by clicking the appropriate button in “No. of words” box. Choosing the association measure for extracting word n-grams is done by clicking the desired measure in “Association mea-

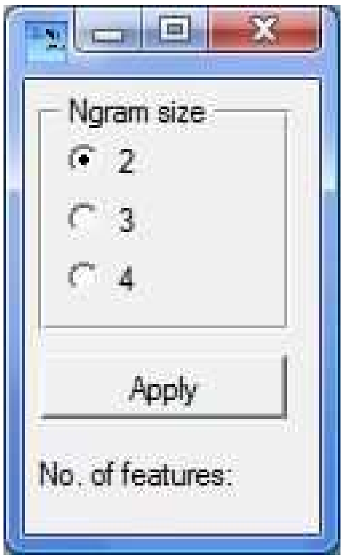


Figure 9.5: Letter n-gram widget

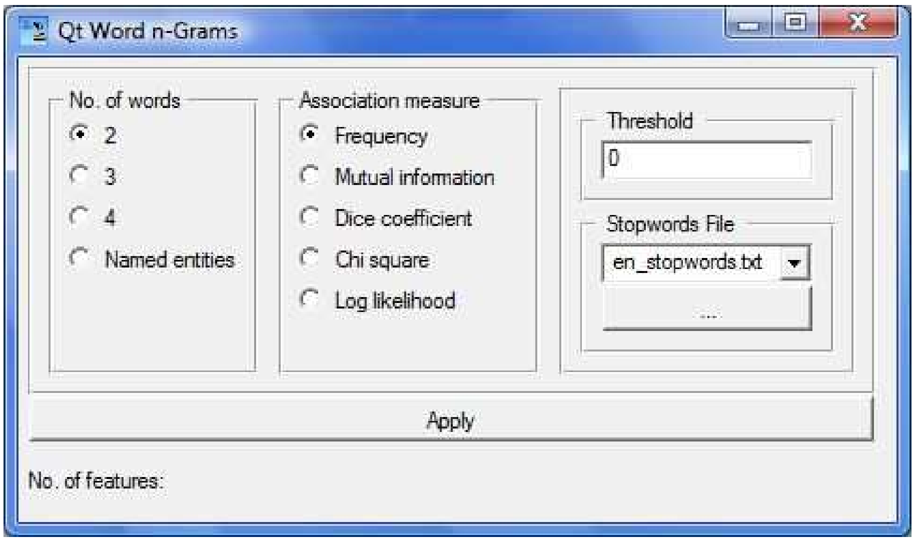


Figure 9.6: Word n-gram widget

sure" box. It is also possible to specify a list of stopwords and set the threshold for the value of the association measure. While this widget also enables extraction of named entities as features ("Named entities" option in "No. of words" box), this option will not be covered here as named entity recognition is outside of scope of this thesis. WordNgram widget is shown in figure 9.6.

9.3.6 TextFeatureSelection widget

Inputs

Examples (ExampleTable)

Outputs

Examples (ExampleTable)

Description

This widget is used for removing some features or documents according to the selected criteria. For a selected measure, we can choose to eliminate those features (documents) that have the value of that measure less than (option "MIN") or more than (option "MAX") a given threshold. Additionally, if the percentage box is checked, the value in the threshold field is interpreted as a percentage. That means that, for example, if we choose TF measure, MIN operator and threshold of 90 with percentage box checked, we are actually eliminating the 90% of features with the smallest value of TF measure (in other words, we are keeping 10% of features with the highest values of TF measure). The measures TF, TDF, RAND, WF, and NF are described in section 9.2 on page 74. Note that it is possible to iteratively apply various selection criteria on the same data. If this data is to be used with the correspondence analysis module, it should be always checked that there are no zero rows. That can be done by applying the following selection criteria: selecting "WF" for measure, "MIN" operator, unckeck "percentage" and use a threshold of 1. That way all documents with zero words will be eliminated. By clicking the "Reset" button, the data is reset to the original state. The TextFeatureSelection widget is shown in figure 9.7.

9.3.7 TextDistance widget

Inputs

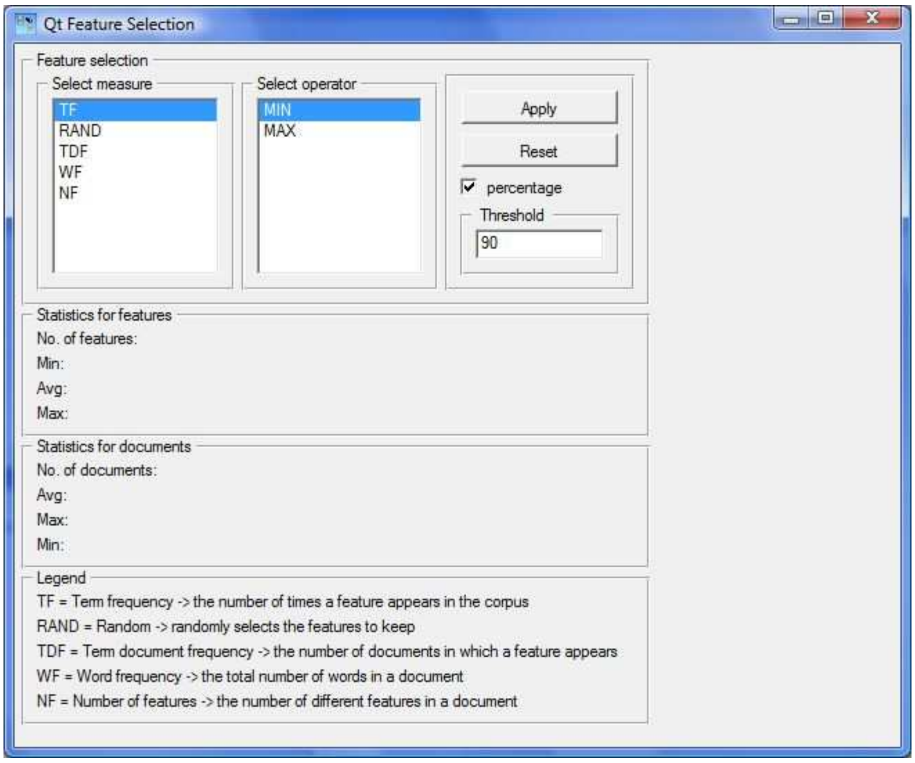
Examples (ExampleTable)

Outputs

Distance matrix (orange.SymMatrix)

Description

The TextDistance widget is very simple. It inputs an ExampleTable with any number of features and computes the cosine between the angle of document vectors in feature space. Its output can be used for any widget that requires a distance matrix. Due to its utter simplicity, this widget will not be shown.



Statistics for features
No. of features:
Min:
Avg:
Max:

Statistics for documents
No. of documents:
Avg:
Max:
Min:

Legend
TF = Term frequency -> the number of times a feature appears in the corpus
RAND = Random -> randomly selects the features to keep
TDF = Term document frequency -> the number of documents in which a feature appears
WF = Word frequency -> the total number of words in a document
NF = Number of features -> the number of different features in a document

Figure 9.7: Feature selection widget

9.3.8 Correspondence analysis widget

Inputs

  Examples (ExampleTable)

Outputs

  (none)

Description

This widget is used to perform the correspondence analysis of an ExampleTable. This widget was implemented by Mladen Kolar. Although some new features have been added, the core functionality of the CA widget is the same as described in [29, §6.1]. A very thorough description of how to use the widget, along with all the options and a few screenshots can be found there. Therefore, this widget will not be dicussed here.

Chapter 10

Comparison of Text Features for Correspondence Analysis

You may never know what results come of your action, but if you do nothing there will be no result

Mahatma Gandhi

Having implemented different text features in Orange, it would be interesting to see how they compare to one another on some text mining task. For this purpose, correspondence analysis of a parallel Croatian-English corpus was chosen. The theory of correspondence analysis was covered in chapter 8. Here it will be shown how well letter n-grams, words, and word n-grams perform on a task of visualization of a corpus. As the corpus is bilingual, results will also be compared for Croatian and English.

Section 10.1 will first explain the criteria that were used to compare the different text features, while section 10.2 will thoroughly describe the results and interpret them.

10.1 Comparison Criteria

For this task, a parallel corpus of newpaper articles from the journal *Croatia Weekly* was chosen [49]. It consisted of 1790 documents in each language (the documents had a one-for-one translation). All the documents were already divided into four categories: politics, economy, tourism and ecology, and culture and sport. Each document belongs to one and only one category.

As was stated in chapter 8, correspondence analysis is an exploratory technique. This means that it gives no definitive numerical results, but an interpretation of the results is needed. It is used by experts to give them an insight about the data they are dealing with. Because of this, the task of comparing different visualizations arising from using different textual features is not an easy one. Some criteria for comparison have to be established. It should be noted that from now on, for simplicity reasons, the term *view* will be used to denote the plot we get by choosing a combination of any two axes, and will be interpreted by an expert from the domain under consideration.

Text features are compared on the following five criteria:

1. For each of the four categories (politics, economy, culture and sport, tourism and ecology), is there at least one combination of axes (one *view*) that separates that category from all others?

2. What is the maximum axis* on which results still make sense?

3. Is there a view that separates all four categories at once?

4. Is there a view that separates sport from culture?

5. Is there a view that separates politics into domestic politics and foreign affairs?

The first criterion is obvious—since every document belongs to one category, one would expect that for each category there exists a view which separates that category from all other.† The second criterion is somewhat unclear—what does it exactly mean that “results make sense?” By this, it is meant that there are still some clusters of documents that are separated from all other, and documents in those clusters are somehow similar to each other. Although this is not a precise definition, neither will the results for this criterion be. Result for the second criterion will be a number which will indicate that the visualizations using axes higher than that number are very poor and that little can be inferred from them based on individual subjective judgement of an expert. The third criterion can be thought of as a stronger first criterion—we not only want that each category is separated from others on some view, we want that there is at least one view that separates all the categories at once. This is like asking correspondence analysis to cluster the data. Since one of the categories is *sport and culture*, it would certainly be interesting to see if the documents within that category are separated from each other based on whether they are about sport or about culture. A similar criterion could be established

*Recall that each subsequent axis accounts for less and less of variance in the data.

†At least, that is what “good” features should be able to do.

for *tourism and ecology*, but that was not done as the documents in that category are very similar (tourism is often interconnected with nature and ecology in general), much more similar than documents about sport and culture. Therefore, even the human interpreting the results had problems with telling tourism and ecology articles apart, so it was decided not to use this criterion. The final criterion of comparison was the ability to separate articles from the *politics* category into those that talk about domestic (Croatian) politics and those that talk about foreign affairs. For anyone with even a little knowledge of politics it was easy to infer if a document is about domestic or foreign politics.

10.2 Results

The following text features are tested: words, letter digrams, letter trigrams, words digrams obtained by frequency, Dice coefficient, mutual information, chi-squared, and log-likelihood, and word trigrams obtained by frequency. The reason for not including word trigrams obtained by other measures is their sparsness. That is, the majority of word trigrams appear in only one or two documents. This means that removing any features by using feature selection (which is necessary as there are much more features than the program can handle) will cause many documents to have no features left. A document without any features has to be removed and so many documents are lost, making the comparison with the previous experiments impossible.

Results for the English part of corpus are given in table 10.1. The columns represent the comparison criteria. The value *somewhat* means that it is hard to say whether or not the given criterion is met, i.e., it depends on the person interpreting the results. When lemmatization was used before adding the features, this is indicated by "(lem)" next to the feature name. Mark "tfidf" next to a feature's name indicates that tfidf value of a feature was used (see equation (9.1)) and not its frequency, while "L2" means that documents were normalized using L2 norm (see equation (9.3)).

Results for the Croatian part of corpus are given in table 10.2. For Croatian, using word trigrams showed to be problematic (in the way it was described earlier in this chapter) even when pure frequency, so those results are not shown. Marks "lem", "tfidf", and "L2" have the same meaning as in table 10.1. Note that even though letter n-grams are usually used without any prior preprocessing, lemmatization was used for letter digrams because the results without lemmatization were so bad that even for the first two axes all the data was in one big cluster, looking like it was randomly plotted. Viewing subsequent axes showed that there was no improvement, so lemmatization

TABLE 10.1: Results for different text features on the English part of corpus

features	1	2 (≈)	3	4	5
words(lem)	yes	9 and 10	somewhat	yes	somewhat
words	yes	9 and 10	somewhat	yes	yes
words (lem, tfidf, L2)	yes	11 and 12	somewhat	yes	yes
letter digrams	no	2 and 3	no	no	no
letter trigrams	no	7 and 8	no	no	yes
word digrams (freq)	yes	13 and 14	somewhat	yes	yes
word digrams (MI)	yes	20 and 21	no	yes	no
word digrams (Dice)	yes	16 and 17	yes	yes	no
word digrams (chi)	yes	10 and 11	yes	yes	somewhat
word digrams (ll)	yes	10 and 11	yes	yes	yes
word trigrams (freq)	yes	15 and 16	somewhat	no	no

TABLE 10.2: Results for different text features on the Croatian part of corpus

features	1	2 (≈)	3	4	5
words(lem)	yes	9 and 10	somewhat	yes	yes
words	yes	5 and 6	somewhat	no	yes
words (lem, tfidf, L2)	yes	10 and 11	no	yes	yes
letter digrams(lem)	no	1 and 2	no	no	no
letter trigrams	yes	6 and 7	somewhat	no	yes
word digrams(lem) (freq)	yes	16 and 17	yes	yes	somewhat
word digrams(lem) (MI)	no	20 and 21	no	yes	no
word digrams(lem) (Dice)	no	16 and 17	yes	yes	no
word digrams(lem) (chi)	yes	10 and 11	somewhat	yes	no
word digrams(lem) (ll)	yes	14 and 15	somewhat	yes	no

was used in an attempt to get any meaningful results.

From tables 10.1 and 10.2, few interesting facts can be observed. First, letter n-grams don't seem to be a very good choice for text features when performing correspondence analysis. This could be due to the fact that letter n-grams are very dense—many different documents (not necesserily belonging to the same categoy) share the same letter n-grams. For example, an often seen problem with letter n-grams was the fact that *tourism and ecology* and *culture and sport* were in the same cluster. Upon inspection, it was found that the n-grams "cul", "ult", "ltu", "tur", and "ure" are very representative for this cluster. The words from which these n-grams came are: *culture, multi-*

culturalism, intercultural, but also *agriculture, horticulture,* and *floriculture.* The first three words are obviously from the documents that talk about culture while the last three are from documents about ecology. When dealing with words and word n-grams, this would never happen.

When comparing different AMs used for extracting word n-grams, it is interesting to notice that Dice and mutual information show somewhat inferior results to chi-square and log-likelihood. This is interesting because Dice and mutual information have been found to give better collocations than chi-square and log-likehood (see the first part of the thesis). Indeed, the features mutual information gave were, for example, *forensic medicine, animated film, genetically modified, national parks, patron saint,* while log-likelihood found n-grams like *previous year, visit Zagreb, percent decrease, particularly important,* which are definitely not collocations (in the sense they were defined in part one). This discrepancy between the results obtained here and those in part one can be explained by taking into account the work done by Chung and Lee [7]. They explored the similarity between mutual information, Yule's coefficient of colligation, cosine, Jaccard coefficient, chi-square, and log-likelihood and what they have found is that, depending on the types of collocations they wished to extract, different measures behaved similarly. In conclusion they state that "it is necessary to select an association measure most appropriate for a given application such as lexical collocation extraction or query expansion because these may need an emphasis on terms in a different range of frequencies". In short, they claim that different measures will behave better for different tasks to which collocations are applied, and the results obtained here seem to support this claim. It is also interesting to compare these results to the work done by Koehn et al. [28]. In one of their experiments they evaluated the use of syntactic phrases (word sequences very similar to the definition of collocation used here) for machine translation, and compared this approach to machine translation using phrases without any restriction. What they found was that not only do syntactic phrases not help the translation, they even harm the translation quality. As an example, they consider the German phrase *es gibt,* which is literally translated as *it gives,* but it actually means *there is.* Of course, *es gibt* and *there is* are not syntactic phrases (nor collocations), so this translation is never learned. Similar examples are phrases *with regard to* and *note that* which are not collocations but have simple one-word translations in German. All this indicates that perhaps for correspondence analysis it is not important that the word n-gram features be collocations, maybe using the n-grams extracted by statistical measures like chi-square and log-likelihood is better.

When comparing word n-grams with just words, for English the word n-grams (using log-likelihood) showed better than words, while this was not the case for Croatian. In fact, word digrams extracted by log-likelihood in the English part of the corpus were the only features that were able to meet all the criteria from section 10.1 (that is, all the criteria except the second one, which cannot be “met”). Some of the plots obtained by using log-likelihood in the English part of corpus are shown in figures 10.1–10.3. Figure 10.1 shows how the documents are grouped into four clusters, corresponding to four categories. In figure 10.2 one long cluster of documents can be seen separating from the other documents—documents in that cluster are all about sport. Separation of domestic and foreign policy can be seen in figure 10.3—the upper blue cluster are documents about foreign policy (they mostly talk about Serbia, Kosovo, Milošević) while the lower blue cluster is made from documents about domestic policy (those documents talk about SDP, HDZ, HNS and other Croatian political parties). Why exactly words perform better than word n-grams for Croatian, and do not for English, is unclear. Obviously, the fact that the two languages are different plays some part in this, but guessing how the difference in the languages is reflected in preferance of different text features for correspondence analysis is outside of scope of this thesis.

How the plot obtained by using same features compares between languages can be seen in figures 10.4 and 10.5.[*] Figure 10.4 shows the first two axes of the plot obtained by using words (lemmatized) as features. It is obvious that the two plots are nearly identical. In contrast, on figure 10.5 it can be seen that when using word digrams (obtained by mutual information) the plots for English and Croatian are different even for the first two axes. The purple cluster spreading to the right, which can be seen in figure 10.5a, is a cluster of documents that are about sport. However, figure 10.5b shows no such cluster (though the purple points that are spreading upwards are documents about sport, there are too few of them to be considered a cluster, and they are also mixed with documents from *tourism and ecology* category).

In comparison of different word features (lemmatized, non-lemmatized, using TFIDF and normalization), it seems that Croatian definitely benefits from lemmatization, while English does not. This is somewhat expected as Croatian is morphologically much more complex than English. Using TFIDF[†] and L2 normalization showed to give similar results as when using just frequency (for Croatian, it even performed slightly worse). However, using TFIDF and normalization yielded some other interesting clusters of documents that were not found using any other features (e.g., a cluster of documents talking about the concentration camp *Jasenovac*, a cluster about mass graves and

[*]These two figures correspond to rows one and seven in tables 10.1 and 10.2.

[†]Note once again that TFIDF values of features were used as input to correspondence analysis—it was not used for feature selection.

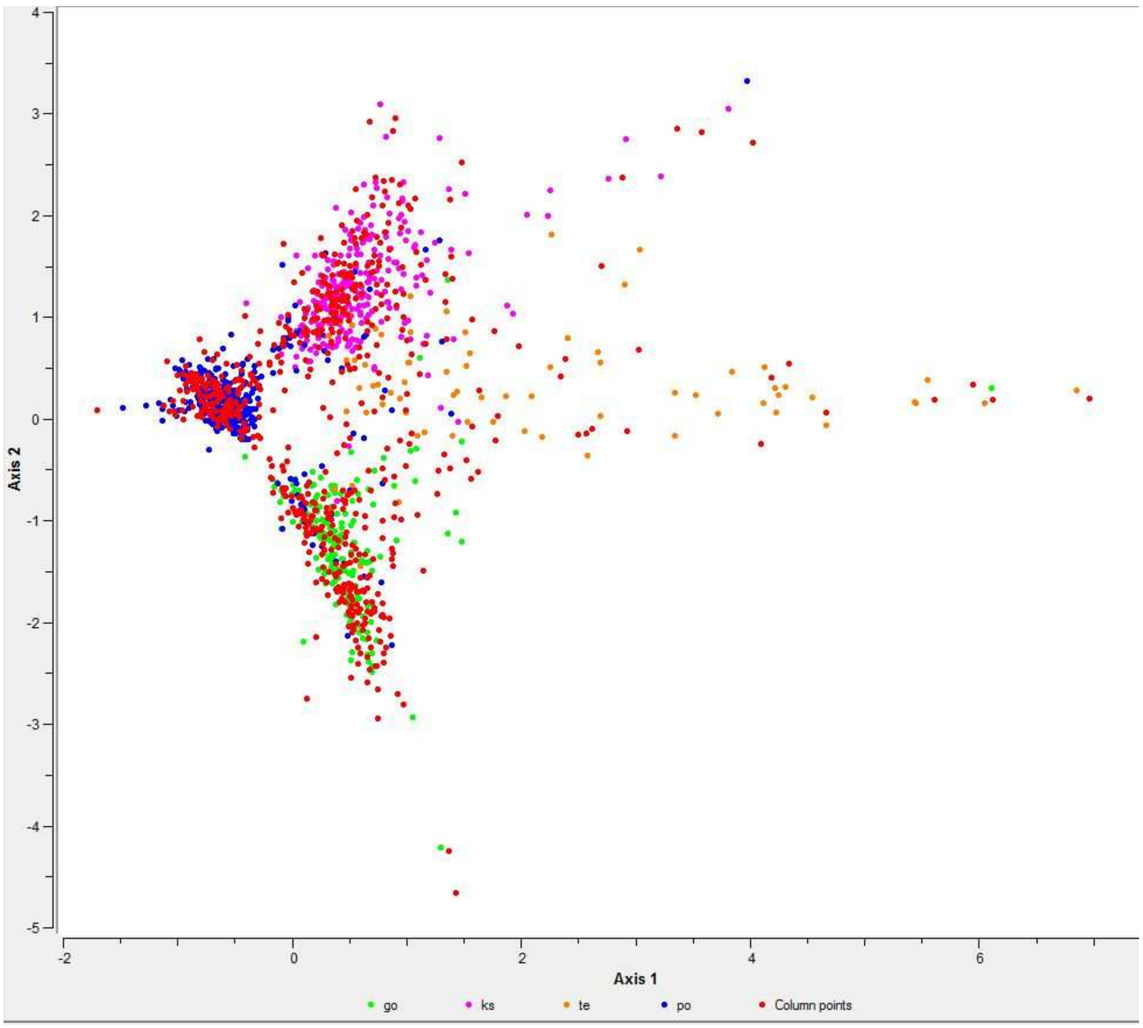


Figure 10.1: Separation of four categories using word digrams on the English part of the corpus. Word digrams are obtained using log-likelihood without lemmatization, so these results correspond to row ten in table 10.1.

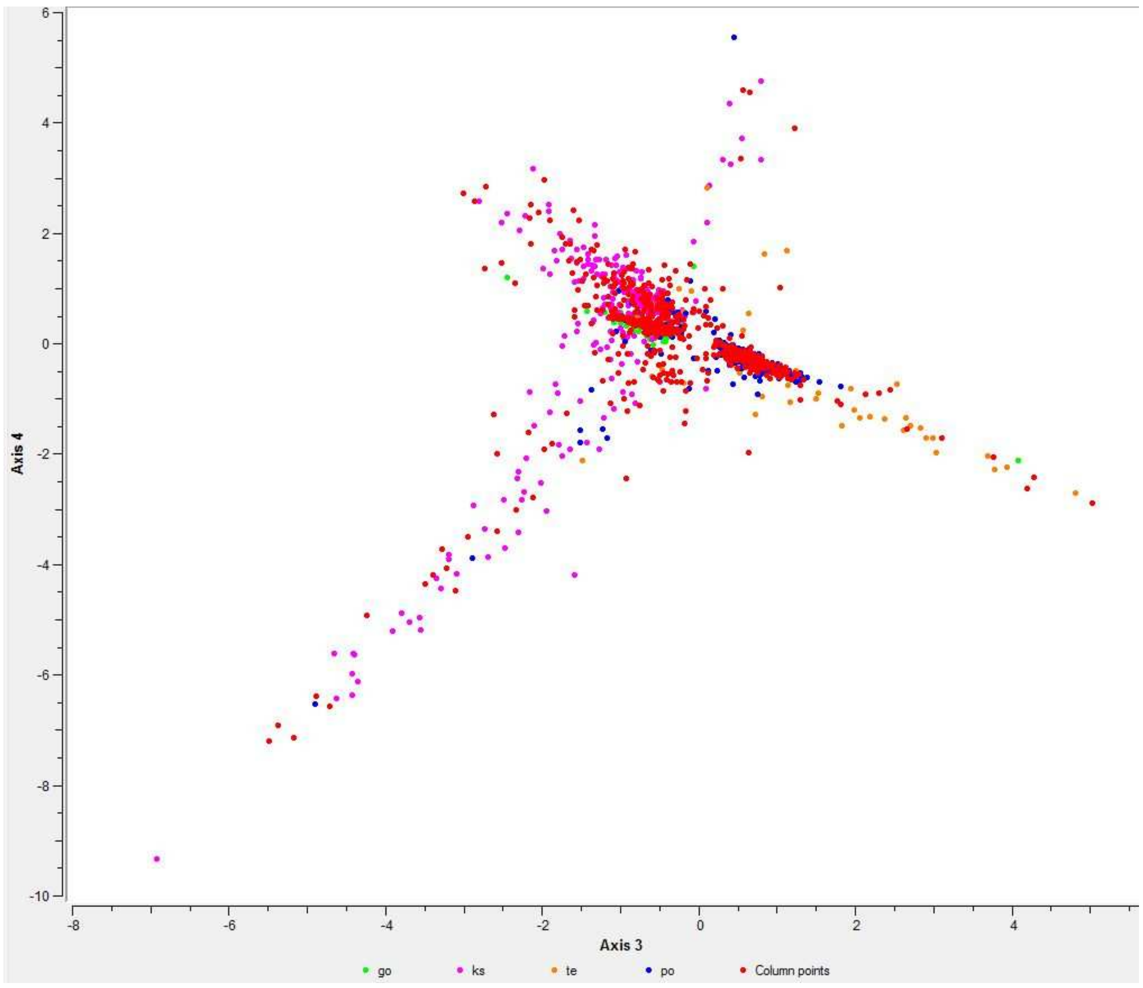


Figure 10.2: Sports cluster as separated by using word digrams (obtained by log-likelihood) on the English part of the corpus. These results correspond to row ten in table 10.1.

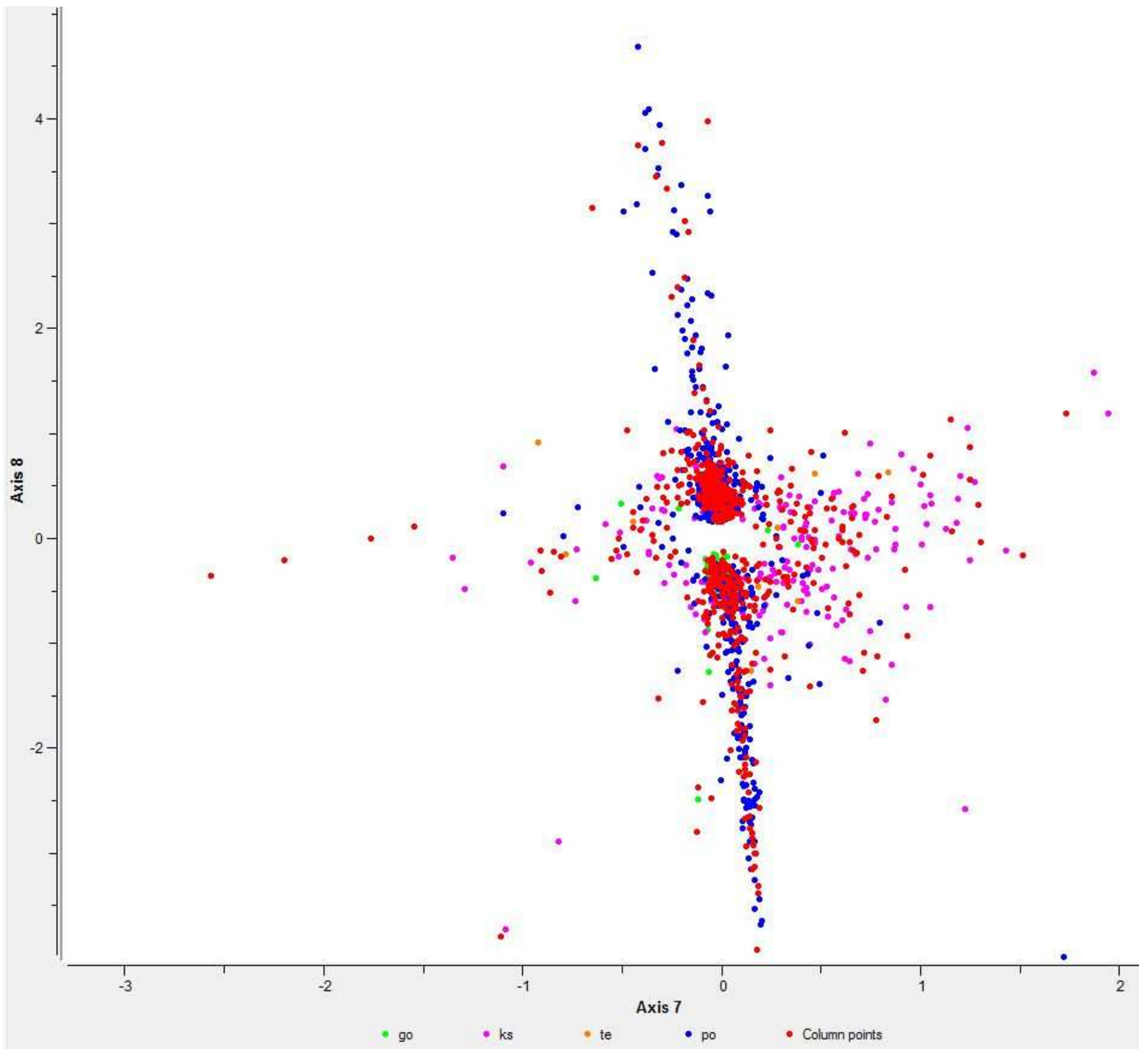
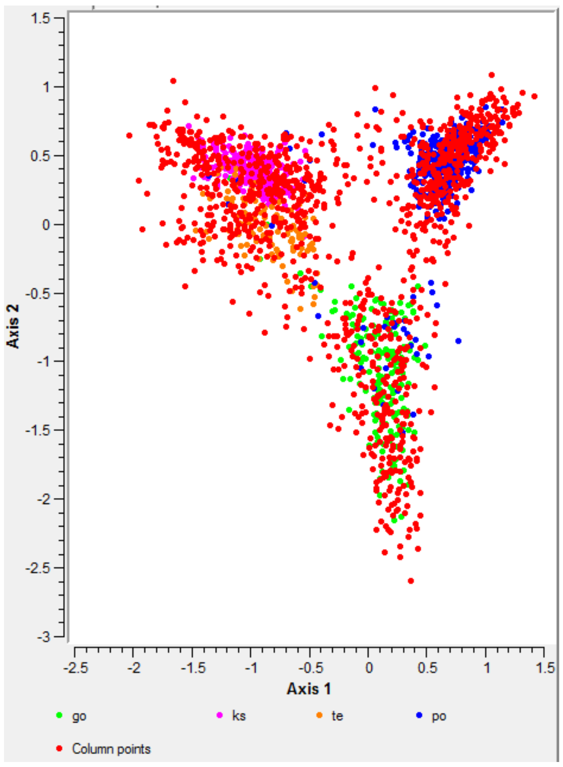
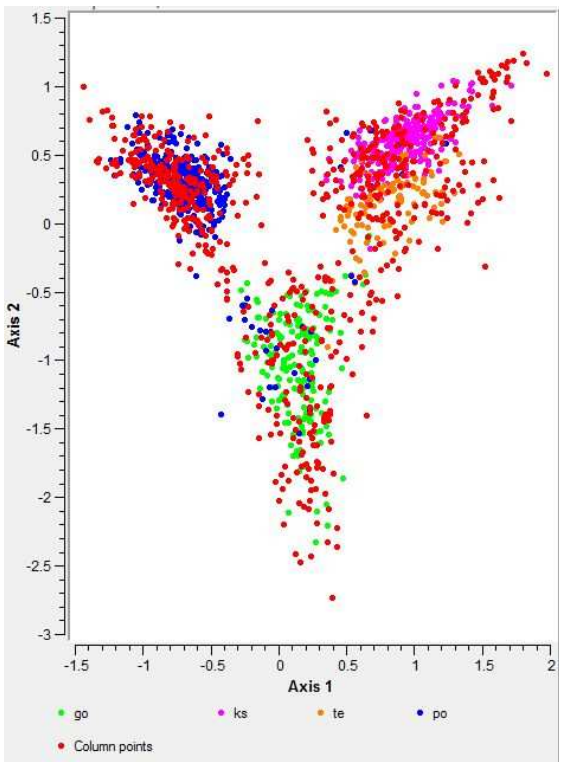


Figure 10.3: Separation of domestic and foregin policy using word digrams (obtained by log-likelihood) on the English part of the corpus. These results correspond to row ten in table 10.1.

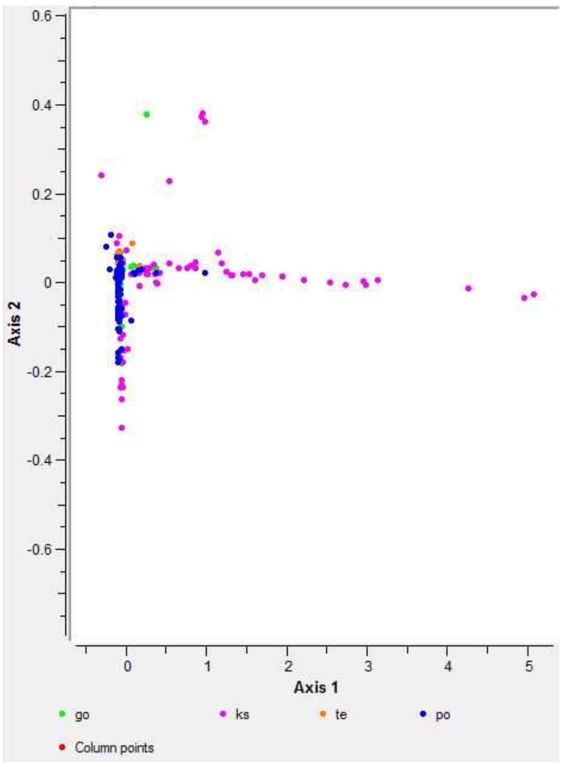


(a) English

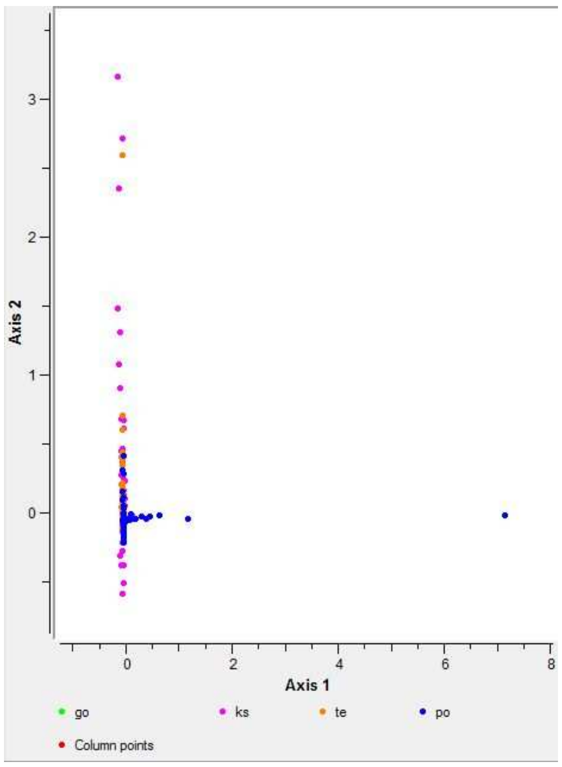


(b) Croatian

Figure 10.4: Plots for two different languages, using words (lemmatized) as features



(a) English



(b) Croatian

Figure 10.5: Plots for two different languages, using word digrams (obtained by mutual information) as features

missing persons during the war in Croatia, etc.). Also, the terms obtained by using TFIDF were much more informative than those obtained by using simple frequency (which is expected as TFIDF chooses the terms that are most relevant for a particular document).

For word n-grams, a series of experiments were conducted that are not shown in tables 10.1 and 10.2. Those experiments included setting different thresholds for AMs, thus keeping "stronger" or "weaker" collocations. Those experiments showed that it was always better to set a lower threshold of an AM and then remove more features using feature selection, than setting a higher threshold of an AM and then remove less features using feature selection. For example, it is better to get 30 thousand word n-grams and then keep the most frequent 5%, thus getting 1500 features, than it is to get three thousand word n-grams and then keep 50% of the most frequent ones (resulting in the same number of final features-1500).

While reading the results, it should be kept in mind that correspondence analysis is only an exploratory technique so any comparison of different representations of one dataset using correspondence analysis is subjective. There can never be any objective comparison criterion for a method that depends on how the results are interpreted. The comparison criteria that were chosen in this case are ad-hoc. Choosing some other criteria for comparison would maybe give different results. However, this is the first time (known to the author) that different text features were compared for visualization of a text corpus using correspondence analysis, and even though the results are somewhat subjective they give a first glimpse on how different text features behave when used in correspondence analysis.

CHAPTER 11

Conclusion of the Second Part

Do or do not... there is no try

Master Yoda

The second part of this thesis deals with application of word n-grams on visualizing a text corpus using correspondence analysis, and with comparing them to some other often used text features on the same task.

Letter n-grams are one of more often used methods for representing text. They are a sequence of n (consecutive) characters extracted from text. Wide use of letter n-grams is due to the fact that they are very easy to implement and do not require any particular linguistic knowledge. Because of the latter, they are language-independent, performing well even for languages like Chinese and Japanese where other traditional methods like the bag-of-words approach fail. Because of their wide use and simplicity, letter n-grams were chosen as one of the features to compare against word n-grams.

The other feature to compare against word n-grams are words. Lemmatized or non-lemmatized, using frequency or TFIDF, words are today probably most frequently used features in text mining. When used without lemmatization, they also require no linguistic knowledge, but unlike letter n-grams they run into problems when dealing with languages like Chinese and Japanese because of the problems with tokenizing words. Still, words as text features have been found to perform very well and have already been exploited in correspondence analysis, making them an ideal candidate for comparing with word n-grams.

Correspondence analysis is a visualization technique used to represent the structure within matrices that contain some measure of correspondence between the rows and columns. It was first proposed by a group of French mathematicians, but it was soon adopted by others as one of the exploratory multivariate techniques and used in many different fields—from linguistics to biology. Correspondence analysis creates a plot of both rows and columns of a matrix in the same space, thus showing the interaction of two categorical variables that are represented by them. Axes are created by correspondence analysis in such a way that each subsequent axis explains less and less of variance in the data. These axes are not interpretabile like in some other techniques, like concept indexing.

Orange is a component-based data mining software. Providing a range of different data preprocessing, modelling, and exploration techniques, Orange's greatest strength is its palette of visualization techniques. The correspondence analysis module was already implemented in Orange, and in this thesis a text preprocessing module for Orange was implemented. The module developed for this purpose, called **orngText**, has the ability to load most of the widley used textual formats, lemmatize the text in two languages (Croatian and English), and extract the already mentioned features (word n-grams, letter n-grams, and words).

Comparing the three different text features was done on a parallel Croatian-English newspaper corpus *Croatia Weekly*. As correspondence analysis is an exploratory technique, some ad-hoc criteria were suggested for comparing the features. Using those criteria, it was shown that letter n-grams do not perform as well as words and word n-grams, and also that word n-grams perform almost the same as words in Croatian, while they perform even better than words in English. The comparison among different association measures also revealed that some measures that performed well in the first part of the thesis did not prove to be as good for use in correspondence analysis. Possible reasons for this discrepancy have also been discussed. As expected, using lemmatization with words showed to improve the results for Croatian, while it did not have much effect in English. The most important contribution of this part of the thesis is that the three text features, word n-grams, words, and letter n-grams, were used and compared in correspondence analysis for the first time.

Abstract

In this thesis, various association measures for extracting collocations from a textual corpus were compared. A lot emphasis was also put on proposing and evaluating different ways of extending the association measures. Extending the association measures is an important issue, but, to this day, only a few authors have addressed it, and there has been almost no effort to compare the different extensions. Different association measures and extension patterns have been compared on three Croatian corpora of different wiriting styles and one English corpus.

The second part of the thesis showed how word n-grams, extracted using the association measures discussed in the first part, can be used for correspondence analysis, a multivariate exploratory technique. They have been compared with words and letter n-grams for the same task. All of the experiments were conducted on a parallel Croatian-English corpus, so results between the two languages were also compared.

Bibliography

[1] Beazly, D. M., *Python Essential Reference, Third Edition.* Sams Publishing, Indianapolis, IN, USA, 2006.

[2] Berthold, M. and Hand, D. J. (editors), *Intelligent Data Analysis.* Springer, 2002.

[3] Biskri, I. and Delisle, S., "Les N-grams de Caractères pour l'Extraction de Connaissances dans des Bases de Donnèes Textuelles Multilingues." In *TALN*, pp. 93–102, 2001.

[4] Blaheta, D. and Johnson, M., "Unsupervised Learning of Multi-Word Verbs." In *Proceedings of the ACL Workshop on Collocations, Toulouse, France*, pp. 54–60, 2001.

[5] Boulis, C., "Clustering of Cepstrum Coefficients Using Pairwise Mutual Information." Tech. Rep. EE516, Electrical Engineering Dept., University of Washington, Seattle, 2002.

[6] Cavnar, W. B. and Trenkle, J. M., "N-gram-Based Text Categorization." In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp. 161–175, Las Vegas, US, 1994.

[7] Chung, Y. M. and Lee, J. Y., "A Corpus-Based Approach to Comparative Evaluation of Statistical Term Association Measures." *Journal of the American Society for Information Science and Technology*, vol. 52(4), pp. 283–296, 2001.

[8] Church, K. and Hanks, P., "Word Association Norms, Mutual Information, and Lexicography." *Computational Linguistics*, vol. 16(1), pp. 22–29, 1990.

[9] Church, K. W., Gale, W. A., Hanks, P., and Hindle, D., "Using Statistical Linguistics in Lexical Analysis." In *Lexical Acquisition: Using On-line Resources to Build a Lexicon*, edited by U. Zernik and L. Erlbaum, pp. 115–165, Hilldale, New Jersey, 1991.

[10] Dagan, I., Church, K. W., and Gale, W. A., "Robust Bilingual Word Alignment for Machine-Aided Translation." In *Proceedings of Workshop on Very Large Corpora: Academic and Industrial Persprectives, Columbus, Ohio*, pp. 1–8, ACL, 1993.

[11] Daille, B., "Combining Symbolic and Statistical Approaches to Language." In *Study and Implementation of Combined Techniques from Automatic Extraction of Terminology*, edited by J. L. Klavans and P. Resnik, chap. 3, MIT Press, Cambridge, MA, USA, 1995.

[12] Damashek, M., "Gauging Similarity with N-grams: Language-Independent Categorization of Text." *Science*, vol. 267(5199), pp. 843–848, 1995.

[13] Demšar, J., Zupan, B., and Leban, G., "Orange: From Experimental Machine Learning to Interactive Data Mining." Tech. rep., Faculty of Computer and Information Science, University of Ljubljana.

[14] Dice, L. R., "Measures of the Amount of Ecologic Association Between Species." *Ecology*, vol. 26(3), pp. 297–302, 1945.

[15] Dunning, T., "Accurate Methods for the Statistics of Surprise and Coincidence." *Computational Linguistics*, vol. 19(1), pp. 61–74, 1994.

[16] Dunning, T., "Statistical Identification of Language." Tech. Rep. MCCS-94-273, Computing Research Lab (CRL), New Mexico State University, 1994.

[17] Evert, S. and Krenn, B., "Using Small Random Samples for the Manual Evaluation of Statistical Evaluation Measures." *Computer Speech and Language*, vol. 19(4), pp. 450–466, 2005.

[18] Fano, R. M., *Transmission of Information; A Statistical Theory of Communication*. MIT Press, New York, 1961.

[19] Fellbaum, C. (editor), *Wordnet: An Electronic Lexical Database*. Bradford Books, 1998.

[20] Goldman, J.-P. and Wehrli, E., "FipsCo: A Syntax-Based System for Terminology Extraction." In *Grammar and Natural Language Processing Conference*, 2001.

[21] Golub, G. and Loan, C. V., *Matrix Computations*. Johns-Hopkins, Baltimore, USA, 1989.

[22] Greenacre, M. J., *Theory and Applications of Correspondence Analysis*. Academic Press, 1984.

[23] Horn, R. A. and Johnson, C. R., *Matrix Analysis*. Cambridge University Press, 1985.

[24] "Hrčak, Portal of scientific journals of Croatia." http://hrcak.srce.hr/.

[25] Jalam, R., *Apprentisage Automatique et Catégorisation de Textes Multilingues*. Ph.D. thesis, Université Lumière Lyon2, 2003.

[26] Johansson, C., "Good Bigrams." In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 96)*, pp. 592–597, 1996.

[27] Kita, K., Kato, Y., Omoto, T., and Yano, Y., "Automatically Extracting Collocations from Corpora for Language Learning." Tech. rep., Dept. of Linguistics, Lancaster University, 1994.

[28] Koehn, P., Och, F. J., and Marcu, D., "Statistical Phrase-Based Translation." In *Proceedings of HLT-NAACL, Edmonton, Alberta, Canada*, pp. 127–133, 2003.

[29] Kolar, M., *Correspondence Analysis*. Diploma thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, 2006.

[30] Kolar, M., Vukmirović, I., Šnajder, J., and Dalbelo Bašić, B., "Computer-Aided Document Indexing System." *Journal of Computing and Information Technology*, vol. 13(4), pp. 299–305, 2005.

[31] Krah, A., Wessel, R., and Pleißner, K.-P., "Assessment of Protein Spot Components Applying Correspondence Analysis for Peptide Mass Fingerprint Data." *PROTEOMICS*, vol. 4(10), pp. 2982–2986, 2004.

[32] Manning, C. and Schütze, H., *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[33] McEnery, A. M. and Oakes, M. P., "Sentence and Word Alignment in the CRATER Project." In *Using Corpora for Language Research*, edited by J. Thomas and M. Short, pp. 211–231, Longman, London, 1996.

[34] McIness, B. T., *Extending the Log Likelihood Measure to Improve Collocation Identification*. Ph.D. thesis, University of Minnesota, 2004.

[35] Miller, E. L., Shen, D., Liu, J., and Nicholas, C., “Performance and Scalability of a Large-Scale N-gram Based Information Retrieval System.” *Journal of Digital Information (online refereed journal)*, 2000.

[36] Morin, A., “Intensive Use of Correspondence Analysis for Information Retrieval.” In *Proceedings of 26th International Conference on Information Technology Interfaces, Cavtat, Croatia*, pp. 255–258, 2004.

[37] “Narodne Novine.” `http://www.nn.hr/`.

[38] Oakes, M. P., *Statistics for Corpus Linguistics*. Edinburgh University Press, 1998.

[39] Pearce, D., “A Comparative Evaluation of Collocation Extraction Techniques.” In *Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002), Las Palmas, Spain*, 2002.

[40] Pecina, P., “An Extensive Empirical Study of Collocation Extraction Methods.” In *Proceedings of the ACL Student Research Workshop*, pp. 13–18, 2005.

[41] Petrović, S., Šnajder, J., Dalbelo Bašić, B., and Kolar, M., “Comparison of Collocation Extraction Measures for Document Indexing.” *Journal of Computer and Information Technology*, vol. 14(4), pp. 321–327, 2006.

[42] Phillips, D., “Correspondence Analysis.” *Social Research Update 7*, 1995.

[43] Porter, M., “An Algorithm for Suffix Stripping.” *Program*, vol. 14(3), pp. 130–137, 1980.

[44] Shannon, C. E., “A Mathematical Theory of Communication.” *The Bell System Technical Journal*, vol. 27(1), pp. 379–423, 1948.

[45] da Silva, J. F. and Lopes, G. P., “A Local Maxima Method and a Fair Dispersion Normalization for Extracting Multi-Word Units from Corpora.” In *6th Meeting on the Mathematics of Language, Orlando, FL*, pp. 369–381, 1999.

[46] Smadja, F. and McKeown, K., “Automatically Extracting and Representing Collocations for Language Generation.” *Proc. of the 28th Annual Meeting of the ACL*, pp. 252–259, 1990.

[47] Šilić, A., Šarić, F., Dalbelo Bašić, B., and Šnajder, J., “TMT: Object-Oriented Text Classification Library.” In *Proceedings of 29th International Conference on Information Technology Interfaces, Cavtat, Croatia*, pp. 559–564, 2007.

[48] Šnajder, J., “Rule-Based Automatic Aquisition of Large-Coverage Morphological Lexica for Information Retrieval.” Tech. rep., ZEMRIS, FER, University of Zagreb, 2005.

[49] Tadić, M., “Building the Croatian-English Parallel Corpus.” In *Proceedings of the LREC 2000 conference, Athens, Greece*, vol. 1, pp. 523–530, 2000.

[50] Tadić, M., “Building the Croatian National Corpus.” In *Proceedings of the LREC 2002 conference, Las Palmas*, vol. 2, pp. 441–446, 2002.

[51] Tadić, M. and Šojat, K., “Finding Multiword Term Candidates in Croatian.” In *Proceedings of IESL2003 Workshop*, pp. 102–107, 2003.

[52] Thanopoulos, A., Fakotakis, N., and Kokkinakis, G., “Comparative Evaluation of Collocation Extraction Metrics.” In *Proceedings of the LREC 2002 Conference*, pp. 609–613, 2002.

[53] “Time.” http://www.time.com/.

[54] Vechtomova, O. and Karamuftuoglu, M., “Use of Noun Phrases in Interactive Search Refinement.” In *Proceedings of MEMURA 2004 workshop*, 2004.

[55] Vechtomova, O., Robertson, S., and Jones, S., “Query Expansion with Long-Span Collocates.” *Information Retrieval*, vol. 6(2), pp. 251–273, 2003.

[56] “Vjesnik, hrvatski politički dnevnik.” http://www.vjesnik.hr/.

[57] Wikipedia, “Singular Value Decomposition—Wikipedia, The Free Encyclopedia.” http://en.wikipedia.org/w/index.php?title=Singular_value_decomposition&oldid=153726159, 2007. [Online; accessed 26-August-2007].

[58] Wu, C.-C. and Chang, J. S., “Bilingual Collocation Extraction Based on Syntactic and Statistical Analyses.” *Computational Linguistics and Chinese Language Processing*, vol. 9(1), pp. 1–20, 2004.

[59] Zamir, S. and Gabriel, K., “Lower Rank Approximation of Matrices by Least Squares with Any Choice of Weights.” *Technometrics*, vol. 21(4), pp. 489–498, 1979.

[60] Zipf, G. K., *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.