D09, D10 & D11

Extracting Domain-specific Collocations for the Dutch WordNet

Date: 25 August 2008

Project: STE05039
http://www.let.vu.nl/onderzoek/projectsites/cornetto/start.htm

Funded by the Stevin framework (www.taalunieversum.org/stevin)

Authors:

- Erik Boiy, Department of Computer Science, Katholieke Universiteit Leuven
- Marie-Francine Moens, Department of Computer Science, Katholieke Universiteit Leuven

Contents

List of Tables

Chapter 1

Introduction

The aim of our work in the Cornetto project was to design and implement a toolkit that aids the lexicographer in creating a lexicon. We have developed this toolkit and applied it for the creation of a domain-specific lexicon (medical and legal domain) for the Dutch language. It offers various functions and parameters to suit different lexicographical needs. In our experiments, we focused on two tasks: the extraction of domain-specific terms and the extraction of collocations. Our methods are purely statistical in nature and learn the lexicographical knowledge from corpora. The reader should keep in mind that the results are highly dependent on the corpora used.

The report is organized as follows. We first give some definitions and describe related research. A next chapter discusses the statistical methods that were developed. We describe the corpora and our experiments. For the usage and implementation details of the software we refer to Appendix A and B respectively.

Chapter 2

Defining collocations

The most specific type of collocations are *idiomatic expressions*; these are fixed word combinations whose meaning cannot be determined from looking to the words in isolation. On the other end of the spectrum we have free-word combinations: words that do not add meaning to each other or their combination, and of which one can easily be replaced without significantly altering the overall meaning. These have no real value for inclusion in a lexicon beyond illustration purposes. In between we find the focus of our research: collocations. It can be hard, however, to properly identify them, as there is a large gray area between these classes. In general, 'collocations cover word pairs and phrases that are commonly used in language, but for which no general syntactic or semantic rules apply'. For semantic collocations[1] the relation between words forming the collocation are defined by a lack of semantic generality, e.g. 'rekening houden met' (to take into account). Grammatical collocations on the other hand consist of an open-class word called the *base* which determines the words it can collocate with, called the *collocators* (often certain prepositions).

Considering the difficulty of *defining* what a collocation is, it has been suggested how to *find* collocations. It is stated by Halliday and Hasan (1976) that collocations are those pairs that occur frequently together in the same environment, but do not include lexical items that have a high overall frequency in language. This approach provides a nice means of finding them by using statistical methods (as we will see in Section 4). Shimohata et al. (1999) similarly defines collocations as 'the way that words occur regularly whenever another word is used'. These broad definitions make evaluation difficult, as in addition to collocations, several other relations that may or may not be interesting to a lexicographer such as (near) synonyms and part-whole relations, fall under the same definition.

McKeown and Radev (2000) note that collocations tend to be tied to specific domains, making them extra valuable when creating a domain specific lexicon, as will be the case in our research. Shimohata et al. (1999) include in their definition of *domain specific* collocations that words must be in a rigid order.

[1]Note that, to add to the confusion, other interpretations of a 'semantic collocation' exists. Sekine et al. (1992) use the term to define 'collocations that reflect ontological relations among entities in given subject domains', used e.g. to disambiguate between the role of 'scarf' ('I saw the girl with the scarf') and 'telescope' ('I saw the moon with the telescope') in relation to the verb 'to see'.

Chapter 3

Related Research

The most basic means of finding how a word behaves is by looking at its context. The so-called 'key word in context' (KWIC) scheme provides the lexicographer with a list of concordance lines centered around the word under investigation. For common words such a large, unordered list is still too unpractical for use. One step beyond KWIC is to extract sequences of words whose frequency of appearance is above a certain threshold. Choueka et al. (1983) tested this approach using 2- to 6-word windows.

The next step, which remains popular, is the use of statistical methods, e.g. a pointwise mutual information statistic (*mi*) (o.a. Church and Hanks 1989; Church et al. 1991). Mutual information compares the probability of observing words A and B together, with the probabilities of observing A and B independently. Likewise they use the t-score as a measure for dissimilarity, e.g. determining that *strong tea* is more likely than *powerful tea*. They use mean and variance between a pair of words to differentiate between fixed (idioms) or more loosely coupled word pairs (collocations). Breidt (1993) applies Church et al. (1991)'s approach to extract German verb-noun combinations. For nouns occurring over 3 times within 5 positions left of the verb (as nouns found here are most likely the verb's object), MI and t-scores were calculated. For further simplification, collocations were only sought for verbs in their infinitive form. The extracted bigrams were manually checked by means of KWIC listings and annotated as being collocations or not. Her approach seems to work well for restricted domains and some interesting conclusions are stated; e.g., lemmatization leads to low precision on an unparsed corpus and larger corpora improve recall with a minor decrease of precision. Lin et al. (1998) also use the MI statistic, but apply it on dependency triples (e.g. verb + its subject, or verb + its noun object) instead of word pairs.

The Xtract system (Smadja and McKeown 1990; Smadja 1993) call their similarity metric (using z-score of a pair) *strength*. The *spread* metric is introduced as the distribution of relative distances between two words. The positions of words from pairs with low spread are flexible; the pairs are considered as having similar syntactic relations called 'predicative relations' (e.g. adjective-noun, adverb-verb, etc.). Low spread may also indicate a semantic rather than a rigid lexical relation of the words (i.e., words that tend to appear in the same context, e.g., 'doctor - nurse'). If defining collocations as lexically constrained pairs, these can be considered candidates for filtering out. In a second phase, Xtract will first look up concordances for the extracted word pairs. Words that appear in the pair's context with sufficient probability are added to the pair to form either phrasal templates (one or more open positions between the correlating words) or rigid compounds (no open positions). Additionally, wrong collocation

word pairs will be expanded to form correct multi-word expressions (e.g. 'blue stocks' becomes 'blue chip stocks') by the same technique.

Seretan et al. (2004) propose a technique to combine collocational bigrams[1] by combining them over *pivot terms*; terms that are identical both as words as in position in the text. The results can then again be combined. Several measures of interestingness are defined to determine the value of the formed collocations, based on frequency, the bigram's collocation scores and loglikelihood. Šnajder et al. (2008) use genetic programming to generate solutions (i.e., formulas) that determine interestingness of three-word collocations. The solutions are represented as a tree. Its leaves can contain constants, frequency information or lexical (part-of-speech (POS)) information. The inner nodes are operators, including binary operators (addition, subtraction, multiplication and division), one unary operator (the natural logarithm), and one ternary operator the IF-THEN-ELSE operator. Yamamoto and Church (2001) use suffix arrays to make the computation of the frequencies of all substrings of a corpus manageable (i.e., for a N-token corpus, the $N(N+1)/2$ possible substrings are grouped in at most $2N-1$ equivalence classes).

Shimohata et al. (1999) explicitly focus on finding domain specific collocations. Their approach has several phases. First "collocation units", which can be strings of any length, are extracted by comparing the entropy of adjacent words preceding and following a string to a certain threshold. A high entropy means high uncertainty w.r.t. predicting the adjacent words, making them poor candidates to form a collocation with. These collocation units are then further combined and filtered based on simple measures as their overlap or subsumption, (relative) frequencies, length (longer is more significant) and positions in the concordance lines. Their approach worked fairly well.

An alternative approach by Pearce (2001) is based on the following definition: 'a pair of words is considered a collocation if one of the words significantly prefers a particular lexical realization of the concept that the other represents'. The technique is based on WordNet synsets. First a collection of candidate collocation synsets for a word w is constructed by taking the synsets that contain a word which co-occurs with w. The most frequent co-occuring term, together with w, is chosen to form a collocation. Collocation strength is then measured as the difference between the frequency of the two top frequent co-occuring words in a synset.

In (Evert and Krenn 2001) a thorough study is made comparing log-likelihood, t-test, chi square, MI and frequency (baseline) techniques, showing the first two performing best overall in their tests. Evert's website[2] contains an overview of many association measures, as well as the UCS toolkit implementing them.

An interesting system is *The Sketch Engine* (Kilgarriff et al. 2008) in which one-page automatic corpus-based summaries of a word's collocational behavior (called a *sketch*) are generated. Grammatical relations are considered, e.g. for a verb, separate collocate lists are presented for the subject, objects, conjoined verbs, modifying adverbs, prepositions and prepositional objects. The system operates by considering lemmas instead of basic words. Interesting for the lexicographer who needs to make a distinction between near-synonyms are the *sketch differences*, which provide lists of shared and non-shared collocations from which he or she can infer the differences in behavior of both words.

Evaluation is usually performed by examining the *top-N* collocation from a ranked lists containing the association measure results. Evert and Krenn (2001) investigate 'snapshots'

[1] Obtained with Goldman's FipsCo

[2] http://www.collocations.de

for different N, perform stepwise processing of the lists to obtain precision and recall graphs and provide separate results for high and low frequency data.

We did not perform any novel research ourselves due to the limited time that we worked on the Cornetto project. Instead we implemented and tested some of the techniques described in this section, offering a generic toolkit for term relation extraction. The relations can take different forms, and the toolkit can be easily ported to process other languages than Dutch.

Chapter 4

Methodology

As seen in the previous section, there are several approaches to the detection of related terms and more specifically collocations, when large corpora are available. We discuss here the techniques that we implemented ourselves. They regard association measures, such as the likelihood ratio for a binomial distribution and a chi-square statistic, both based on hypothesis testing, and the pointwise mutual information statistic. In addition, we discuss frequency measures such as frequent item sets.

4.1 Likelihood ratio

Following (Dunning 1993), we test the hypothesis of independence between a term (i.e. a word or a multi-word expression) and a "concept", being either another term or a certain class (e.g. the financial law domain), with the likelihood ratio for a binomial distribution. If we can reject the hypothesis, there is a strong indication that the word is correlated with the concept.
We first define p_1 and p_2:

$$p_1 = P(term = t|concept = k) \qquad p_2 = P(term = t|concept! = k)$$

The hypotheses of independence (H_1) assumes that p_1 and p_2 are equal, i.e.:

$$H_1 : p_1 = p_2 = p$$

whereas H_2 allows all possible values of p_1 and p_2. We define the likelihood functions given a contingency table (Table 4.1). The cells of this table indicate the number of occurrences; c_{12} is the number of occurrences of term t together with concept k, c_2 is the total number of occurrences of t and c_1 is the number of occurrences of k. N is the total number of terms in the used corpora.

$$L(H_1) = b(c_{12}; c_1, p)b(c_2 - c_{12}; N - c_1, p)$$
$$L(H_2) = b(c_{12}; c_1, p_1)b(c_2 - c_{12}; N - c_1, p_2)$$

where $b(k; n, x)$ is the binomial distribution. The maximum value of the likelihood functions above is obtained by setting their first derivative to zero, which yields for p, p_1 and p_2 the following values:

$$p = \frac{c_2}{N} \qquad p_1 = \frac{c_{12}}{c_1} \qquad p_2 = \frac{c_2 - c_{12}}{N - c_1}$$

	concept = k	concept != k	
term = t	c_{12}	$c_2 - c_{12}$	c_2
term != t	$c_1 - c_{12}$	$N + c_{12} - c_1 - c_2$	$N - c_2$
	c_1	$N - c_1$	N

Table 4.1: Contingency table for words appearing in the visual and non-visual corpus.

We then determine the likelihood ratio λ

$$\begin{aligned} \lambda &= \frac{L(H_1)}{L(H_2)} \\ &= \frac{L(p, c_{12}, c_1)L(p, c_2 - c_{12}, N - c_1)}{L(p_1, c_{12}, c_1)L(p_2, c_2 - c_{12}, N - c_1)} \end{aligned}$$

with

$$L(p, k, n) = p^k(1-p)^{n-k}$$

We take

$$\begin{aligned} \log \lambda = & -\log L(p_1, c_{12}, c_1) - \log L(p_2, c_2 - c_{12}, N - c_1) \\ & +\log L(p, c_{12}, c_1) + \log L(p, c_2 - c_{12}, N - c_1) \end{aligned}$$

which is a value that is asymptotically χ^2 distributed. By selecting a confidence level from the χ^2 distribution table (1 degree of freedom), the obtained likelihood ratio value allows us to accept or to reject the H_1 hypothesis.

4.2 Chi-square

Given the above contingency table we can test the hypothesis of independence of the occurrence of a term and a concept by means of a χ^2 test. The χ^2 statistic is defined as:

$$\chi^2 = \sum_{i,j} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

with $E_{i,j}$ being the expected value for $O_{i,j}$ (the observed value, found at position i, j in the cells of the contingency table). The expected frequencies $E_{i,j}$ are calculated from the marginal probabilities:

$$E_{i,j} = \frac{O_{i,1} + O_{i,2}}{N} \times \frac{O_{1,j} + O_{2,j}}{N} \times N$$

The resulting value is χ^2 distributed and allows us to accept or reject independence as was the case for the likelihood ratio. In order to obtain reliable statistics, it is required that $N > 50$ and $O_{i,j} > 5$. According to (Dunning 1993), the likelihood ratio for a binomial distribution more reliably copes with sparse data.

4.3 Point-wise mutual information statistic

A pointwise mutual information statistic mi quantifies the association strength between two events. It essentially measures the amount of information one event x gives about another event y where $p(x)$ denotes the probability that x occurs, and $p(x, y)$ that both events occur jointly.

$$mi(x,y) = log\frac{p(x,y)}{p(x)p(y)}$$

The pointwise mutual information compares two models (using Kullback-Leibler (KL) divergence) for predicting the co-occurrence of x and y. One model is the maximum likelihood estimate (MLE) of the joint probability of x ad y, and the other is the MLE of x and y occurring independently. The pointwise mutual information is high, when x and y occur together more often than they would be by chance, which is computed by the probability of x and y occurring independently.

4.4 Frequent itemsets

The association of a term with a specific class or corpus can simply be based on its frequency of occurrence in a particular corpus, possibly compared to its frequency in a general corpus. When looking for frequent combinations of words, (e.g. as to form a collocation) frequent item set is an efficient way to proceed. The extraction of *frequent itemsets* extraction is a subproblem of association rule mining (see e.g. Agrawal and Srikant 1994). A frequent itemset consists of items that have a support above a user-defined minimum. We define "support" as the number of text units (in our case sentences) that contain all items (words) in the set, upon the total number of sentences. Free software for frequent item set mining is widely available, we used an implementation by Borgelt[1] of the Apriori algorithm. The algorithm functions bottom-up, i.e. it will construct frequent itemsets by attempting to extent an itemset by selecting one item at a time. It searches breadth first and uses a hash tree structure for counting. The main strength of the algorithm is its efficiency.

[1] See http://www.borgelt.net//software.html.

Chapter 5

Experiments

5.1 Overview

For our experiments we focus on two tasks: the extraction of domain specific words, and the extraction of collocations. For the latter we will consider collocation bigrams, n-grams of a larger size, and collocations matching syntactic templates. We test these tasks on corpora from the medical and financial legal domains. This chapter is organized as follows. First we describe the corpora on which we performed our tests, followed by a small description and the test results for each subtask. The results mentioned here are only illustrative, complete results for all techniques are separately available as a deliverable and evaluated.

5.2 Corpora

One domain-specific corpus (for the extraction of collocations), or a domain-specific and general corpus (for finding domain specific terms) are needed to apply the association measures from the previous section. If we would like to find domain-specific collocations, the availability of a domain-specific and general corpus is advisable. We collected the following corpora (all in Dutch):

- **Medical corpus:** contains the text of the Merck Manual and the medical encyclopedia from Spectrum[1], and the articles from Wikipedia classified under relevant subcategories of "Geneeskunde".[2] We used these data when testing because of the availability of a "ground truth" collection of domain specific words, and the easier interpretability (due to the sources included) of the results. The corpus contains about 2.5 million tokens.
- **Financial law corpus:** obtained from EURLex[3] by collecting all documents having the EUROVOC keyword "finances". The corpus contains about 27 million tokens.
- **Wikipedia corpus:** consists of the Dutch Wikipedia pages. A part of the medical corpus stems from the same source, the articles intersecting with the medical corpus were removed. The corpus contains about 65 million tokens.

[1]Note that the material of the Spectrum encyclopedia is in principle not available for free use. We used these data for internal research purposes; for other applications a licence with Spectrum may have to be agreed.

[2]See http://www.merckmanual.nl, http://www.kiesbeter.nl/medischeinformatie/Medischeencyclopedie and http://nl.wikipedia.org.

[3]European Union law, see http://eur-lex.europa.eu

The corpora are lemmatized using TadPole[4]. All symbols and numbers other than punctuation are ignored for the software's operations. We will not use the full corpora but instead use subsets of an indicated size, as using smaller corpora did, at first glance[5], not produce worse results and considerably reduced computation time.

5.3 Extraction of domain specific terms

For the domain specificity we follow an approach in the line of (Damle and Uren 2005). We use association measures (AM) (see Section 4) to determine which words are related to the specific domain, represented by a *target corpus* (TC), compared to a general *reference corpus* (RC). We illustrate here with results obtained with the χ^2 statistic (see Table 5.3). Results obtained by all methods are included in the delivered results.

5.3.1 Results

Test setup: As the TC we used the Financial law corpus, as RC the Wikipedia corpus. We limited both corpora to about 10 million words to reduce computation.[6] The association metrics score the words in TC to produce a ranking of domain specific words. The 10 highest ranked results for nouns, adjectives and verbs using the χ^2 statistic (removed decimals) are given in Table 5.1.

Output: We supply as output lists of nouns, adjectives and verbs specific to the financial legal domain, obtained using the χ^2 statistic, likelihood ratio for a binomial distribution, pointwise mutual information statistic and frequency. The lists are sorted from most to least domain specific. The tail of the list contains words only slightly related to the financial legal domain.

(a) Nouns

Lemmas	χ^2 **value**
artikel	45723
commissie	43111
verordening	32042
lidstaat	21421
raad	16041
gemeenschap	14126
lid	12470
richtlijn	11501
maatregel	10709
bedrag	10566

(b) Adjectives

Lemmas	χ^2 **value**
europees	26374
financieel	21179
overeenkomstig	11452
communautair	10035
gemeenschappelijk	6935
economisch	5997
nationaal	5693
totaal	5219
bevoegd	5070
laatstelijk	3912

(c) Verbs

Lemmas	χ^2 **value**
vaststellen	15373
moeten	10637
overwegende	9487
wijzigen	8751
dienen	7987
bedoelen	7220
letten	7196
betrekken	5310
verstrekken	5114
betreffen	4935

Table 5.1: Nouns (a), adjectives (b) and verbs (c) specific to the financial legal domain as determined by the χ^2 statistic.

It is immediately apparent from the results that the association metrics highly rank words that are strongly associated with one corpus. Thus many words are not from the financial

[4]See http://ilk.uvt.nl/tadpole/.

[5]We have no gold standard for collocations, so it remains an educated guess.

[6]The actual number diverts slightly because the last sentence before reaching the limit is read completely and due to various filtering.

law domain, but have a lot to do with European Union law, or with the formal document style. Therefor we performed a small additional test.

Test setup: As the TC we used the Financial law corpus, as RC the EURLex texts to which the keyword "politics". We limited both corpora to about 35,000 words[7]. The 10 highest ranked results for nouns using the χ^2 measure are given in Table 5.2.

Output: We supply the output for the above test obtained by the χ^2 measure.

(a) Politics (EURLex)

Lemmas	χ^2 **value**
bank	175
tegenpartij	161
activa	151
transactie	149
eurosysteem	133
lijst	76
bedrag	63
rentevoet	59
schuldbewijs	58
gegeven	54

(b) Wikipedia

Lemmas	χ^2 **value**
bank	204
tegenpartij	170
activa	160
transactie	157
eurosysteem	140
lidstaat	117
commissie	90
bedrag	82
rentevoet	62
schuldbewijs	62

Table 5.2: Nouns specific to the financial legal domain using a sample of EURLex (politics) (a) or Wikipedia (b) corpus, as determined by a χ^2 statistic.

The results of the setups using different RC are very similar and we didn't think it was worth creating an additional corpus to pursue this path, although some words related to European Union law (lidstaat, commissie) are not present in 5.1(a), so it helps to some extent.

5.3.2 Evaluation

Test setup: As the TC we used the Medical corpus, as RC the Wikipedia corpus, using 2.5 million words, of which we selected nouns and adjectives. We evaluated their correctness against an online medical lexicon[8], containing both popular en scientific terminology. We only evaluated words that appear both in the lexicon and in our corpus. Evaluation is done in terms of precision and recall for the top N results, were N goes from 50 to 5,000. The results are given in Table 5.3.

Note that this evaluation is tentative, as the lexicon also contains terms that are related to science in general, or terms that are common in general language (and thus not specific to the medical domain). We noticed also that the used medical lexicon is far from complete (e.g., it contains "baarmoeder" (uterus) but not "nier" (kidney)). We see that here the χ^2 statistic performs the best. The highest ranked words that are considered correct or incorrect, respectively. are present or not present in our lexicon:

- *Present:* antibioticum, lichamelijk, contact, bijwerking, verschijnsel

[7]Downloading the documents needs to be semi-manually performed.
[8]Obtained from http://users.ugent.be/~rvdstich/eugloss/NE/lijst.html.

- *Not present:* week, nier, bloeding, operatie, nodig

N	Random		Frequency		λ statistic		χ^2 statistic	
	P%	R%	P%	R%	P%	R%	P%	R%
50	12.00	0.31	30.00	0.77	32.00	0.83	30.00	0.77
100	17.00	0.88	24.00	1.24	23.00	1.19	29.00	1.50
250	16.00	2.06	31.60	4.07	30.00	3.87	34.40	4.44
1000	16.10	8.30	30.30	15.63	28.70	14.80	32.10	16.55
2500	13.92	17.95	24.84	32.03	20.64	26.61	25.20	32.49
5000	12.08	31.15	19.38	49.97	13.22	34.09	19.10	49.25

Table 5.3: Nouns specific to the financial legal domain using a sample of EURLex (politics) (a) or Wikipedia (b), as determined by a χ^2 statistic.

The method for determining domain specific words has also been applied for the detection of visual or non-visual entities and attributes, as described in *Boiy E, Deschacht K and Moens M F. Learning Visual Entities and their Visual Attributes from Text Corpora. Accepted for TIR-08.*

5.4 Extraction of collocations

As shown above, there is no single definition of a collocation. We give here the definition of Cruse (1986): "sequences of lexical items which habitually co-occur, but which are nonetheless fully transparent in the sense that each lexical constituent is also a semantic constituent... they do have a kind of semantic cohesion – the constituents are mutually selective." In practice our system will extract multi-word units, ranging from free-word combinations that often co-ccur, to fixed idioms.

A first basic functionality of our software is the generation of so-called "key word in context" lines; a list of all contexts in which the key word appears. From these lines, the lexicographer can manually spot word combinations he thinks fit the collocation requirements. This is a tedious process which we attempt to automate. Our system, like most in literature, operates in two steps. The first handles the extraction of candidate collocations, the second filters the candidates for relevance, where relevance is often defined in terms of a syntactic template to be followed. We implemented various support functions (see Appendix B for a complete overview).

5.4.1 Candidate collocation filtering and extraction

This area of functionality will look for possible word combinations. Each word can be tagged with its lemma and POS class. Inclusion of collocation candidates may be bound by several possible factors:

- The words occur within a predefined distance (called the *window*) from each other. There is an option whether word order should be respected.
- The presence of a certain required base word for the collocation candidate, or absence of a blacklisted word (e.g. a stopword or a word with undesired POS class).

- The words match a required syntactic template (see examples below). An option indicates whether the order in which the constituents are matched may vary.

Syntactic templates can span more than 2 words. In this case, collocation bigrams are searched as the first step as before, and then further combined[9] to form combinations matching the desired template. To do this we implemented an algorithm based on (Seretan et al. 2004) and extended it with the necessary filters.

5.4.2 Collocation ranking

In this stage, various association measures (see Section 4) can be applied on the gathered co-occurence statistics, allowing for a ranking of the collocation candidates. A cutoff is then used to determine which of these are to be considered true collocations. This cutoff can signify the rejectance of an independence hypothesis, or may be chosen more arbitrarily.

5.4.3 Extraction of domain specific collocations

The following approach can be taken:

1. Extract a list L of domain specific words using hypothesis testing by comparing TC and RC.
2. Form collocations with top N extracted words, including either one or more words from L and the other from TC.

However, as the lexicographer will normally be interested in the collocational behavior of a selected word, these two steps will be used separately, using human input to select the words from L to find collocations for. As our experiments showed, the test for domain specificity add little value, as extracted collocations tend to be already specific to the corpus they were extracted from.

5.4.4 Results

In the next sections we give results ranked by frequency of occurrence as they are easy interpretable and don't differ much from the rankings produced by e.g. the likelihood ratio measure. Frequency is considered to be a good approach to collocation extraction (see e.g. Evert and Krenn 2001). However it has the disadvantage that collocations that rarely occur are not detected, where in this case the hypotheses testing techniques might yield better results. The results show words in their lemmatized form.

Test setup: The first test extracts collocation bigrams (the applied window size is shown in parenthesis); we selected adjective-noun (2), noun-verb (4) and verb-preposition (2) combinations from the Medical (Table 5.5) and Financial Law corpus (Table 5.4), respecting word order when counting collocation candidates. For verb-preposition bigrams, the combinations including "zijn" en "worden" ("to be" and "to become") are ommitted. Corpus size was about 1 million tokens.

[9] This is done by merging the bigrams over a shared word (meaning it needs to be at the same position in the same sentence).

Output: In addition to the results illustrated above above, adverb-verb combinations are included. Techniques for which results are supplied are the χ^2 statistic (for which collocations with a frequency under 3 are discarded, because they tend to dominate the results), likelihood ratio for a binomial distreibution, pointwise mutual information statistic and frequency.

We performed an additional test were at least one of the words in the bigram needed to be among the top 500 ranked domain specific words (determined by the test of Section 5.3). Results showed very little difference in the top ranked results, and the top 10 was identical. Therefore, these results are not illustrated here.

(a) Adjective-noun

Bigrams	**Frequency**
europees gemeenschap	838
bevoegd autoriteiten	525
europees officier	523
centraal bank	521
europees unie	432
europees parlement	426
overeenkomstig artikel	344
intern markt	259
financieel belang	225
forfaitair invoerwaard	157

(b) Noun-verb

Bigrams	**Frequency**
rekening houden	369
gemeenschap letten	344
wijzigen verordening	307
dienen dekking	297
krediet dienen	259
letten verdrag	251
volgen verordening	219
toelichting dienen	215
treden werking	205
artikel treden	188

(c) Verb-preposition

Bigrams	**Frequency**
wijzigen bij	438
houden met	380
vervangen door	343
dienen ter	283
leiden tot	157
maken van	148
alcohol-volumegehalte van	131
voldoen aan	129
gelden voor	121
gebruiken voor	102

Table 5.4: Top 10 lists of collocation bigrams, taken from the financial law corpus and ranked w.r.t. frequency.

Test setup: The second test extracts collocations of unlimited size using a bigram combination technique. First, all closest bigram components within a given window are extracted (using nouns, adjectives, verbs, adverbs and prepositions). A cutoff was applied for the bigrams in order to be valid for the combination; only the ones considered statistically significant by our χ^2 measure with certainty of 99.5% were used to form combinations. The longest valid combination of the bigrams is selected in case of collocations that subsume each other. About 1 million tokens of the financial law corpus were used as input. We give here some examples in the next paragraph of collocations consisting of 4 or more words (with constituent POS tags):[10] that were ranked highest w.r.t. their frequency of occurrence.

"verordening/N verbinden/WW onderdeel/N rechtstreeks/ADJ toepasselijk/ADJ lidstaat/N", "artikel/N verordening/N treden/WW werking/N", "toelichting/N krediet/N dienen/WW

[10]Note that more combinations with less words appear high in the output.

(a) Adjective-noun

Bigrams	Frequency
hoog bloeddruk	149
rood bloedcel	145
dik darm	120
wit bloedcel	112
dun darm	110
aangeboren aandoening	109
erfelijk materiaal	108
lang tijd	108
groot hoeveelheid	91
verschillende soort	90

(b) Noun-verb

Bigrams	Frequency
vragen hebben	340
hebben vragen	337
brief hebben	329
lezen brief	329
volgen contact	324
contact terugkomen	321
behandeling zoeken	262
bellen erfolijn	262
last hebben	215
nemen contact	195

(c) Verb-preposition

Bigrams	Frequency
veroorzaken door	504
lezen van	330
leiden tot	276
gebruiken bij	268
mailen naar	262
bestaan uit	230
komen voor	198
hebben van	190
ontstaan door	181
zorgen voor	164

Table 5.5: Top 10 lists of collocation bigrams, taken from the medical corpus and ranked w.r.t. frequency.

ter/VZ".
A next test aims to improve results by requiring that the collocations match a fixed syntactic template.

Test setup: This test extracts collocation trigrams; we selected noun-verb-preposition and noun-preposition-noun templates from the Financial Law corpus (Table 5.6) and Medical (Table 5.7) corpus. Trigrams are constructed by first extracting all closest bigram components as in the previous test. Bigrams are then combined only if they fit the template. The template used are (window size is again shown in parenthesis): noun-verb (4), verb-preposition (2) for the first template, and noun-preposition (1) for the second, i.e. requiring the words to be adjacent in the last template. The results are sorted w.r.t. their frequency of occurrence. Corpus size was about 1 million tokens. We also included results where each collocation includes a word a priori selected.

In Table 5.6 you can also find combinations for two words for which a lexicographer identified the collocations. Here we used a corpus size of 10 million tokens. This was sufficient for combinations with the second word, but not for the combinations with the first. For "schuldquote" (debt ratio) she found "schuldquote ramen op" and "schuldquote komt_uit op", which are not found by our method (the noun-verb parts are present with a frequency of 1, with a different preposition). For "accijns" (excise) she identified the collocations "vrijstelling van accijns" and "tarief van accijns". The first is successfully extracted, the second is generally found in the corpus as "accijnstarief", or tarief may be used in the compound "minimum-/maximumtarief".

(a) Noun-verb-preposition

Trigrams	Frequency
rekening houden met	289
krediet dienen ter	258
toelichting dienen ter	199
tabel dienen ter	104
artikel vervangen door	90
lid vervangen door	84
overeenstemming zijn met	81
gebruik maken van	77
verordening wijzigen bij	69
verband houden met	64
schuldquote dalen tot	7
schuldquote verwachten tot	4
schuldquote blijven onder	2
schuldquote dalen van	2
schuldquote oplopen van	2
schuldquote stijgen van	2
schuldquote worden voor	2
schuldquote bevredigen tot	1
schuldquote beïnvloeden door	1
schuldquote blijven naar	1

(b) Noun-preposition-noun

Trigrams	Frequency
officier van justitie	549
artikel van verordening	302
verdrag tot oprichting	297
lidstaat van herkomst	148
zin van artikel	128
grond van artikel	103
aanbieder van betalingsdienst	101
artikel van richtlijn	101
gebruiker van betalingsdienst	94
procedure van artikel	94
vrijstelling van accijns	30
voorstel voor accijns	14
schorsing van accijns	13
beschikking van accijns	8
accijns van sigaret	7
accijns van tabaksfabrikaat	6
evaluatie door accijns	5
indiening van accijns	5
artikel met accijns	4
heffing van accijns	4

Table 5.6: Top 10 lists of collocation trigrams matching the given template, taken from the financial law corpus and ranked w.r.t. frequency. The second part of the tables has the requirement that a certain word is present. For (a) this is "schuldquote" (debt ratio), for (b) is "accijns" (excise).

Output: The full output of the tests for which the results are displayed in Tables 5.6 and 5.7 is included.

A final test was performed by using frequent itemset detection in sentences, where collocations are composed of frequent combinations of words. This might be interesting for finding large expressions commonly taking up most of a sentence. With the above bigram gram combinations, we already detect frequent item sets, however, other algoritms such as expansions of the a priori algorithm could be tested. We performed some initial tests with the a priori algorithm, but had to abandon this path due to time constraints and not too promising initial results.

5.4.5 Evaluation

We deliver all the outputs of our algorithms. A more in depth evaluation of our output will be performed by domain experts.

(a) Noun-verb-preposition

Trigrams	**Frequency**
last hebben van	155
geneesmiddel gebruiken bij	134
last krijgen van	70
gevolg zijn van	55
contact opnemen met	50
rol spelen bij	50
ziekte veroorzaken door	45
gebruik maken van	38
aandoening komen voor	35
sprake zijn van	35

(b) Noun-preposition-noun

Trigrams	**Frequency**
informatie over diagnose	262
keer per dag	89
vorm van kanker	68
verloop van tijd	67
persoon tot persoon	59
behulp van enzym	37
keer per week	36
gebrek aan vitamine	28
liter per dag	28
risicofactor voor hart	27

Table 5.7: Top 10 lists of collocation trigrams matching the given template, taken from the medical corpus and ranked w.r.t. frequency.

Chapter 6

Acknowledgements

Thanks to Isa Maks for fully annotating a small sample of financial law terms and providing a definition for collocations suited for the project's needs. Thanks to the rest of the Cornetto team for their support.

References

Agrawal R and Srikant R (1994). Fast algorithms for mining association rules. In: Bocca J B, Jarke M and Zaniolo C, eds., Proc. 20th Int. Conf. Very Large Data Bases, VLDB. Morgan Kaufmann, pp. 487–499.

Breidt E (1993). Extraction of v–n–collocations from text–corpora: A feasibility study for german. In: Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives. Association for Computational Linguistics.

Choueka Y, Klein S and Neuwitz E (1983). Automatic retrieval of frequent idiomatic and collocational expressions in a large corpus. Journal of the Association for Literary and Linguistic Computing, 4:34–8.

Church K, Gale W, Hanks P and Hindle D (1991). Parsing, word associations and typical predicate-argument relations. In: Tomita M, ed., Current Issues in Parsing Technology. Kluwer Academic, Dordrecht, Netherlands.

Church K W and Hanks P (1989). Word association norms, mutual information, and lexicography. In: Proceedings of the 27th. Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Vancouver, B.C., pp. 76–83.

Cruse D A (1986). Lexical Semantics. Cambridge University Press, Cambridge.

Damle D and Uren V (2005). Extracting significant words from corpora for ontology extraction. In: K-CAP '05: Proceedings of the 3rd international conference on Knowledge capture. ACM, New York, NY, USA, pp. 187–188.

Dunning T E (1993). Accurate methods for the statistics of surprise and coincidence. Computational Linguistics, 19:61–74.

Evert S and Krenn B (2001). Methods for the qualitative evaluation of lexical association measures. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics. Toulouse, France.

Halliday M A K and Hasan R (1976). Cohesion in English (English Language). Longman Pub Group.

Kilgarriff A, Rychlý P, Smrž P and Tugwell D (2008). The sketch engine. In: Practical Lexicography: A Reader. pp. 297–306.

Lin S H, Shih C S, Chen M C, Ho J M, Ko M T and Huang Y M (1998). Extracting classification knowledge of internet documents with mining term associations: A semantic approach. In: Research and Development in Information Retrieval. pp. 241–249.

McKeown K R and Radev D R (2000). Handbook of Natural Language Processing, chapter 21. Marcel Dekker, Inc., New York, NY, USA.

Pearce D (2001). Using conceptual similarity for collocation extraction. In: Proceedings of the Fourth UK Special Interest Group for Computational Linguistics (CLUK4). pp. 34–42.

Sekine S, Carroll J, Ananiadou S and Tsujii J (1992). Automatic learning for semantic collocation. In: Proceedings of the Third Conference on Applied Natural Language Processing. Association for Computational Linguistics, pp. 104–110.

Seretan V, Nerima L and Wehrli E (2004). Multi-word collocation extraction by syntactic composition of collocation bigrams. In: Nicolov N, Bontcheva K, Angelova G and Mitkov R, eds., Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003, Current Issues in Linguistic Theory. John Benjamins, Amsterdam/Philadelphia, pp. 91–100.

Shimohata S, Sugio T and Nagata J (1999). Retrieving domain-specific collocations by co-occurrences and word order constraints. Computational Intelligence, 15:92–100.

Smadja F (1993). Retrieving collocations from text: Xtract. Computational Linguistics, 19:143–177.

Smadja F A and McKeown K R (1990). Automatically extracting and representing collocations for language generation. In: Proceedings of the 28th. Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Pittsburgh, PA., pp. 252–259.

Šnajder J, Dalbelo Bašić B, Petrović S and Sikirić I (2008). Evolving new lexical association measures using genetic programming. In: Proceedings of ACL-08: HLT, Short Papers. Association for Computational Linguistics, Columbus, Ohio, pp. 181–184.

Yamamoto M and Church K (2001). Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. Computational Linguistics, 27:1–30.

Appendix A

Usage

A.1 Setting up the software

- Simply copy the jar file to the location of choosing. Make sure you have java 1.5.0 or higher installed.
- Preferably install TadPole and its dependencies, see http://ilk.uvt.nl/tadpole/. Note that TadPole operates with ISO 8859-15 (or a close variant), while corpora are supplied in UTF-8. The supplied reader for TadPole output will use the ISO 8859-15 encoding.

A.2 Running the software

Type

java -jar collocations.jar

followed by the command of choice and the necessary parameters.

Quite a lot of information is sent to the console. As there is no "verbose" switch implemented, it may be a good idea to pipe the console output to some temporary file (using > *tempfile*) once you get things right.

A.2.1 Commands

Below commands are given for performing specific tasks. For an overview of the parameters see the next section. Note that filtering and cutoff options are not included here as not to make the commands bloated, but can nevertheless be included.

Find domain specific words:
domspec -tc *path* -rc *path* [-size *S*] [-l] [-counts *path*] -am *am* -out *path*

Find collocation bigrams:
bigram -tc *path* [-size *S*] [-l] [-pairs *"pos1 pos2 w"*] [-fixedOrder] [-counts *path*] -am *am* -out *path*

Find collocations matching (at least) one of the given templates (referred to as *patterns* in the software):
pattern -tc *path* [-size *S*] -l [-pairs *"pos1 pos2 wA" "pos2 pos3 wB"*] [-patterns *"pos1 pos2*

pos3"] [-counts *path*] -am *am* -out *path*

E.g. to obtain the results from Table 5.7 for the "N WW VZ" template, type:
pattern -out hm -tc *lemmatized_corpus_location* -size 1000000 -l -am frequency -pairs "WW VZ 2" "N WW 4" -verbfilt -patterns "N WW VZ"

A.2.2 Parameters

The parametrization is not complete, but contains the most important options.[1] Advanced users will want to interface the code through the API. Note that although the code allows for other POS taggers, this interface works only for TadPole output. If a nullpointer is encountered, most likely you didn't specify all necessary parameters. If POS classes are required for a parameter, they are taken literally: they are the prefixes of the tags assigned by TadPole. E.g., "WW" = verb and "VZ" = preposition. See the TadPole documentation for tag meanings.

The input parameters:

-tc *path* Specifies the location of the target corpus, or the corpus from which to extract collocations.

-rc *path* Specifies the reference corpus location. *Only used by* ***domspec*** *command.*

-size *S* Specifies the maximum number of (valid) words read by the corpora used.

-l Flag indicating that the input is lemmatized (by TadPole).

Filters:[2]

-inclw* *word1 word2 ...* Include only collocations containing the given words.

-exclw* *word1 word2 ...* Exclude results containing the given words.

-inclwf#* *path* Include only collocations containing one of the first # words found in file at the given location. The file should contain a simple list of words. One word, delimited by a whitespace character or the end of the line, is taken from each line.

-exclwf#* *path* Exclude results (words/collocations) containing one of the first # words foud in the file at the given location. The file should contain a simple list of words. One word, delimited by a whitespace character or the end of the line, is taken from each line.

-inclpos *pos1 pos2 ...* Include only tokens with one of the given POS classes. *Only used by* ***domspec*** *command.*

-verbfilt Will (for the most part) remove auxilliary verbs from verb-filtered results by changing their tags within a window of 5 words preceding a past particle (or infinitive).

Obtaining results:

[1] Note that the reduced number of options may cause some minor differences with the results given in this report.

[2] A "*" must be replaced by the letter indicating the token representation, see "Output parameters" > **-rep**.

-pairs ***"pos1 pos2 window"*** Will cause the extraction of pairs with given template and window. May supply multiple pairs between quotation marks. *Not used by* ***domspec*** *command.*

-patterns ***"pos1 ... posN window"*** Will cause the extraction of pairs matching one (ore more) of the given templates. Note that the **-pairs** command must be used in such a manner that the given templates can be constructed from the given pairs. May supply multiple templates between quotation marks. *Only used by* ***pattern*** *command.*

-fixedOrder Flag indicating that the order of occurrence of POS classes in supplied templates must be respected. *Only used by* ***bigram*** *command.*

-defwindow ***size*** The default window size, used when no POS restrictions are given. Default is 4.

-counts ***path*** The location of the statistics file. If present (i.e. it was obtained by a previous run) the statistics are reused, if it does not exist it will be created containing the statistics of this run. Be careful if using this option, if other parameters are changed, the statistics file is most likely no longer compatible (parameters in this block are safe to change). This option is useful when different AMs are applied to the same data.

-am ***am*** The AM to use. Options are frequency, relfreq (relative frequency – use for domain specificity tests), chisquare, likelihoodlog and mi (mutual information).

-cutoff ***c*** The cutoff to apply to the results in terms of AM score. Default is 0.

-n ***N*** The cutoff to apply to the results in terms of absolute number of results. Default: no limit applied.

Output parameters:

-rep ***char*** The internal and output representation of a token. "T" = token as it appears in the input, "t" = lowercased token, "l" = the token's lemma, "p" = lemma augmented with its POS tag. For the last two the **-l** needs to be used. The default is "p" when the **-l** option is used, otherwise **-t**.

-out ***path*** The location to store the results output file.

Appendix B

Architecture

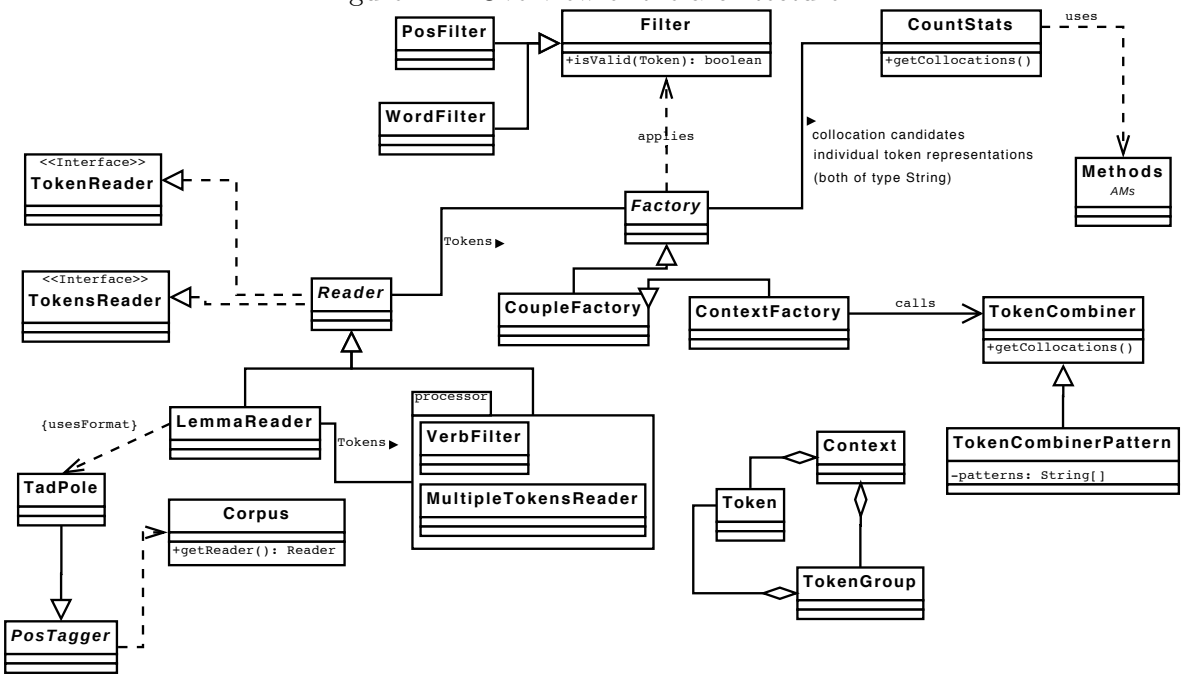
Figure B.1 provides a high level (and incomplete) overview of the architecture, covering only the most important points to keep it compact. More information can be found in the code's documentation. The package structure should prove intuitive for the most part. Our entry point into explaining the architecture is the Corpus class. This class contains information about the corpus formatting (e.g. is it tokenized, or which POS tagger was used to process it). The method createReader() uses this information to return a suitable Reader. The Reader will feed the corpus data into the program, either word by word (TokenReader interface) or one sentence at a time (TokensReader interface). The Reader class has several subclasses to handle unprocessed corpora, POS tagged corpora (by either *Mbt3* or *LtPos*), lemmatized corpora (*TadPole*). A PosTagger subclass contains the necessary information about to POS tagger's output format. The LemmaReader class was hardcoded for use with *TadPole* and will also read POS tags. This is the Reader we used for our testing. While code for corpora in other formats is supplied, they have not been used since the early stages of development and it is recommended to process your corpora using TadPole. A final type of Reader subclass are "processors", which take input from another Reader class object and apply some transformation. The TokensMapper class will map variations of several Tokens onto a single Token, as specified by an input file describing the mapping. This can e.g. by used to map transitive verbs to their infinitive form. More useful is the VerbFilter, which will discard supporting verbs occurring within a given window of a past particle (or infinitive form). A MultipleTokensReader can combine (subsections of) corpora.

The Tokens then serve as input for a Factory class, which extracts information from which to draw statistics. Subclasses of Factory include the TokenFactory for single Tokens and the CoupleFactory for the extraction of collocation bigram candidates. The latter can select all couples within a given window, or only those that match best. There is a parameter indicating if word order should be kept. Like all Factories, it comes with (optional) Filters which puts a limit to the amount of information output. Filters can be applied to a Token, determining its validity. Subclasses include the PosFilter, which requires a Token to be of a certain allowed POS class. The WordFilter states which words are required or should be ignored. The option can for instance be used to find the collocational behavior of a certain word, or to exclude stopwords. A global parameter indicates how tokens should be represented when they appear in Factory output as Strings.[1]

The CountStats class is the final stage for the extraction of domain specific words

[1]This is a means of mapping several variants to a single representation, a.o. increasing available statistics.

Figure B.1: Overview of the architecture.



(CountStatsCross) or collocation bigrams (CountStatsSingle). As the name indicates, this class collects all statistics (i.e. word and collocations counts) needed to apply the AMs discussed (and additional ones that were not discussed). The AMs are contained in the Methods class. Finally, after applying the AM of choice, getCollocations() returns collocation-score or domainword-score pairs which can be sorted and printed.

An additional step is needed when extracting collocations of length > 2. To this end, in a second pass over the corpora, the ContextFactory class will for a given Context (in this case a sentence) add all previously extracted collocation bigrams present in that Context. To further process each Context, it calls a TokenCombiner to combine the context's collocation bigrams by the method discussed earlier to form larger collocations (each collocation is stored in a TokenGroup object). From the TokenCombiner the constructed collocations can then finally be obtained. Two scoring options are supplied: adding the scores of the constituent collocations, or using the frequency of the finally formed collocation. The TokenCombiner-Pattern class works in the same manner, but requires the formed collocation to match one of the supplied patterns.

Additional classes exist, e.g. to obtain corpora or a KWIC breakdown of domain specific words. Check the software's documentation for usage.