Automatic Thesaurus Generation in the Legal Domain using Word Embeddings

Joris van Vugt*

January 15, 2017

Abstract

Legal professionals often use a specialized thesaurus in search engines to retrieve relevant documents. Constructing and maintaining a thesaurus is a laborous task. Past efforts in automatic thesaurus generation relied on knowledge databases or very complicated pipelines. In this paper, I propose a simple model which uses Word2Vec to learn word relationships in an unsupervised manner on a large, freely available corpus of Dutch legal documents. The proposed method outperforms a similar method using latent semantic analysis.

1 Introduction

The goal of the project is to develop a model for automatic thesaurus generation for legal documents. Thesauri are used by legal professionals to aid the extraction of relevant documents with specialized search engines. Manually building and maintaining a thesaurus is a labor-intensive process. Automatic thesaurus generation is thus very useful, but often comes at the cost of lower quality. The problem is intrinsically hard, since specialized thesauri often contain infrequent words which have to be related to other infrequent words. Past methods for automatic thesaurus construction often required external knowledge. Recently, *Word2Vec* algorithms have been very popular. They are unsupervised and allow "arithmetic" with words (e.g., $king - man + woman \approx queen$) [1]. This property is very useful for automatic thesaurus generation, since it allows for reasoning about relations between words, rather than just similarities. In this project, I will propose and validate an approach for thesaurus generation using Word2Vec. My research question is:

Can Word2Vec improve the quality of automatically generated thesauri using no external knowledge?

Past work relied on already existing lexical databases such as WordNet to construct thesauri [2]. Other work used sophisticated *natural language processing* (NLP) pipelines to establish word relations [3, 4]. Hodge & Austin [5] have devised a method which constructs a thesaurus from raw text with no prior knowledge. In a sense, it is similar to the Word2Vec algorithm used here, since it uses a neural network and a sliding window over the text. However, their method is much more specialized and less general. *Latent Semantic Analysis* (LSA) was often used to compute word similarities prior to the conception of Word2Vec [6, 7]. LSA will therefore serve as a baseline in this study.

The dataset consists of 300.000 Dutch public legal documents in XML format retrieved from rechtspraak.nl. Using such a specialized dataset will ensure that it contains most of the relevant words for the domain. A downside of this dataset is that it also contains many words which are irrelevant for a legal thesaurus, because the cause of the lawsuit is explained in each document. A legal thesaurus provided by the Dutch government is used as ground

*s4279859 – jorisvan.vugt@student.ru.nl

Metric	Count
Number of entries	5,558
Number of terms	13,606
Number of unique terms	7,339
Mean number of terms per entry	2.45

Table 1: Statistics about the legal thesaurus used by the Dutch government which is used as ground truth in this study.

truth[1].

Section 2 will provide a brief introduction to the methods used in this study. The results are described in section 3 and will be interpreted in section 4, where I will also give some pointers for further research. Lastly, I will conclude my findings.

2 Methods

2.1 Preprocessing

Since the dataset is in XML format, the raw text first has to be extracted. I have achieved this by simply removing all XML-tags. In addition, all words were converted to lowercase and all interpunction was removed. However, this still leaves some noise in the data, such as document meta-data (e.g., identifier, format, etc.), miscellaneous unicode characters and encoding errors (e.g., HTML entities). Because of the size of the dataset and later processing, the noise shouldn't affect the result too much.

Next, lemmatization and multi-word-unit detection is applied using Frog [8]. Lemmatization means converting each word to its lemma (e.g. *"kings"* becomes *"king"*). This is useful to reduce the size of the vocabulary. Moreover, different conjugations of a word are not useful in a thesaurus. Multi-word-unit detection entails detecting common phrases and converting them to a single "word" by replacing spaces with underscores (e.g., *"buckingham palace"* becomes *"buckingham_palace"*. Thesauri often contain short phrases, because they still refer to a single concept.

[1] https://data.overheid.nl/data/dataset/justitiethesaurus-2015

2.2 Thesaurus Generation

Both Word2Vec and LSA produce a vector representation for each word in the vocabulary. I will first describe both algorithms and then explain how these vectors can be converted to a thesaurus.

2.2.1 Latent Semantic Analysis

A classic approach to finding word similarity is *Latent Semantic Analysis* (LSA). This is typically done using *Singular Value Decomposition*, which showed to provide the similarities best interpreted by humans [7] when compared with *Non-Negative Matrix Factorization* (NMF) and *Latent Dirichlet Allocation* (LDA).

First, tf-idf features are computed for each document

$$\text{tf-idf}(t,d) = (1 + \log \text{tf}(t,d)) \times \text{idf}(t) \quad (1)$$

$$\text{idf}(t) = \log\left(\frac{1+|C|}{1+\text{df}(t)}\right) + 1 \quad (2)$$

where $\text{tf}(t,d)$ is the term frequency (using sublinear scaling), $|C|$ the total number of documents and $\text{df}(t)$ the number of documents t appears in. The resulting matrix is normalized using the euclidean norm. All terms which occur in less than 20 documents were discarded as these are likely noise and irrelevant. Terms which occur in more than 95% of the documents were also removed. (i.e. stop-word removal). Note that this is a bag-of-words approach. In other words, no positional information of words is used.

Next, SVD is applied to the resulting tf-idf matrix. SVD factorizes a matrix into 3 smaller matrices

$$X = U\Sigma V^T \quad (3)$$

where the columns of U and V respectively represent the left-singular and right-singular

vectors. Σ is a diagonal matrix with non-zero values on the diagonal. The columns in all matrices are ordered by their importance. Therefore, if only the first t columns are computed, we get some lower dimensional approximation of the original matrix. This is called *truncated SVD*. I have used the implementation from sklearn [9] to produce 100 dimensional vectors for each word. Because of time constraints, a subset of 100,000 documents was used.

2.2.2 Word Embeddings

Word2Vec [1, 10] is a class of algorithms which can be used to find word embeddings. In other words, each term is assosciated with an n-dimensional vector in latent space, where n is typically much smaller than the vocabulary size. Word2Vec makes the assumption that words that appear in similar context have similar meaning. Note that similar meaning is very broad in this case and can mean synonym, antonym or a related term in general. I have applied the *skip-gram* algorithm with *hierarchical softmax*, which were shown to provide the best results for large corpera. In the skip-gram architecture, a neural network is given a word and has to predict the words that come before and after it (See Figure 1). A more thorough explanation of the architecture can be found in [11]. I have used the original C implementation[2] to generate 300-dimensional vectors using a skip-gram window of 10. For this purpose, all 300,000 files were concatenated into one large text file. Terms which occurred less than 50 times were discarded.

2.2.3 Constructing the Thesaurus

The thesaurus is constructed by first reducing the size of the vocabulary to that of the thesaurus serving as the ground truth. Note that this way the quality of the thesaurus can only be high as the ground truth. Next, the pairwise distance between each word is computed. In the case of Word2Vec, the cosine similarity is used (higher is better) as suggested in [10]. In the case of LSA, the euclidean distance is used (lower is better) which provided better results in [7].

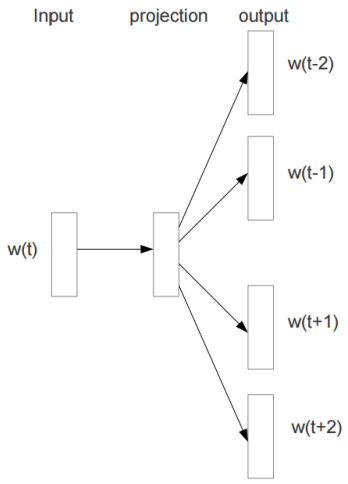


Figure 1: The skip-gram architecture with a window of 4. Given the word in the middle of the window, the model has to predict the words that come before and after it. Both the parameters of the network and the word embeddings are optimized in this task. Figure is reproduced from [1].

$$\text{Cosine-Similarity}(a, b) = \frac{a \cdot b}{\|a\| \times \|b\|} \tag{4}$$

$$\text{Euclidean-Distance}(a, b) = \|a - b\| \tag{5}$$

Now, for each word in the vocabulary we find all the words that are sufficiently similar and add it to the thesaurus. Ideally, this is done by finding some threshold, so that words can have varying numbers of related words. However, I have chosen to use the top-n words instead.

2.3 Evaluation Metrics

The constructed thesaurus is compared to the ground truth and the classic performance metrics from information retrieval can be com-

[2] https://code.google.com/archive/p/word2vec/

puted

$$\text{Recall} = \frac{|r \cap t|}{|r|} \tag{6}$$

$$\text{Precision} = \frac{|r \cap t|}{|t|} \tag{7}$$

where r is the set of words that are deemed similar by the algorithm and t is the set of words that are in the ground truth thesaurus. Recall measures the fraction of relevant words that are retrieved and precision measures the fraction of retrieved words that are relevant. Recall and precision are averaged over all words in the thesaurus.

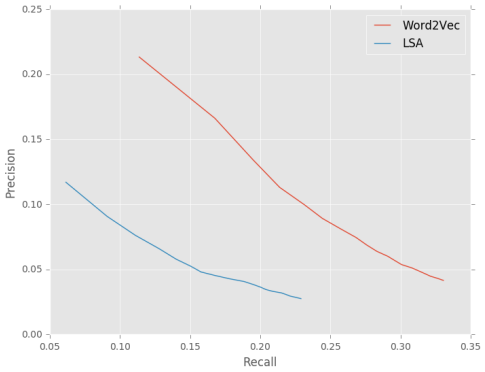


Figure 2: Precision-recall curve. The number of similar terms is varied from 1 to 20. Word2Vec clearly outperforms LSA for each setting.

3 Results

LSA yields a vocabulary of 112,613 terms which is reduced to 2,599 terms after cross-referencing it with the ground truth vocabulary. Similarly the size of the vocabulary of Word2Vec is 109,671 which is reduced to 2,506. In comparison, the ground truth thesaurus consists of 5,558 entries and 7,339 unique terms in total. Both algorithms have thus managed to capture a decent fraction of the words which are relevant. The terms in the ground truth were not lemmatized, which slightly decreases the performance of both models.

Table 2 shows a sample of the related terms generated by the Word2Vec model compared to the terms in the ground truth.

Figure 2 shows the precision-recall curve achieved by varying the number of similar words from 1 to 20. Word2Vec strictly outperforms the LSA approach. In other words, regardless of the application (recall-oriented or precision-oriented) Word2Vec seems to be the most sensible choice. A naive approach of setting the fixed number of similar words is to choose the mean of the number of similar words in the ground truth thesaurus. The mean number of similar words is 2.45 in the dataset. Setting the number of similar words to 2 gives a recall of 0.168 and a precision of 0.166 for the Word2Vec approach (compared to 0.091 precision and recall for LSA).

3.1 Visualizing the Word Embeddings

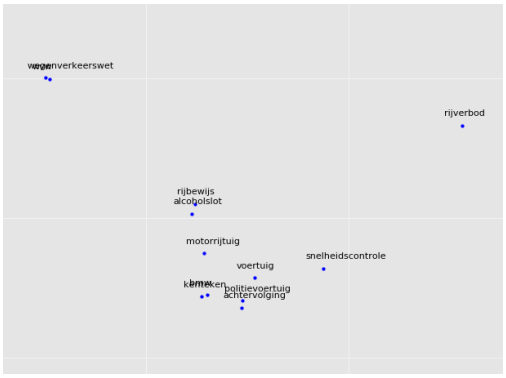
Using *t-Distributed Stochastic Neighbour Embedding* (t-SNE) [12] the 300-dimenensional word vectors can be reduced to two dimensions. This allows us to plot the representation in latent semantic space of each word. Figure 3 shows a few clusters found in the 2-dimensional embedding. Every area in the embedding is easy to interpret, which proves that the embeddings are of high quality.

4 Discussion

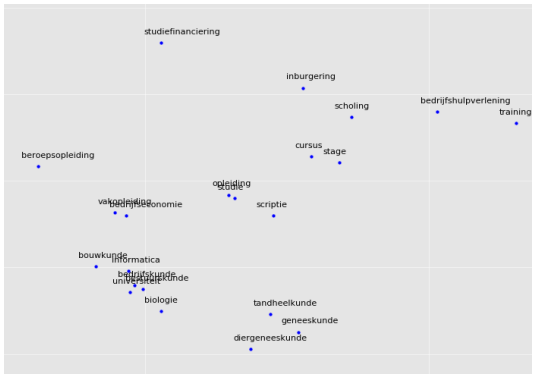
Using Word2Vec for automatic thesaurus generation with no external knowledge drastically improves on similar methods using LSA. Word2Vec is also more promising for more complicated tasks, such as distinguishing between broader terms, narrower terms and synoyms. This is because semantic relations like these often emerge in the word embeddings produced by Word2Vec. It could be as simple

Word	related: Word2Vec	related: ground truth
afbetaling	huurkoop	koop op afbetaling
	koop	consumentenkrediet
	consumentenkoop	huurkoop
afluisteren	telefoontap	opnemen
	geheimhoudersgesprek	aftappen
	aftappen	richtmicrofoon
	observatie	telefoontap
	opsporingsonderzoek	opsporingsmethode
	geluidsband	geheimhoudersgesprek
moord	doodslag	gewelddelict
	medeplichtigheid	levensdelict
	verkrachting	seriemoord
	poging	eerwraak
	uitlokking	crime passionnel
	medeplegen	roofmoord
	mishandeling	liquidatie
	afpersing	cold case
	levensdelict	doodslag
rechtszekerheid	evenredigheidsbeginsel	vertrouwensbeginsel
	legaliteitsbeginsel	algemeen rechtsbeginsel
	rechtspleging	terugwerkende kracht
	vertrouwensbeginsel	rechtsverwerking
uitbuiting	mensenhandel	vrijheidsberoving
	prostitutie	mensenhandel
	dwang	zwarte arbeid
vaderschap	afstamming	afstamming
	adoptie	erkenning van kinderen
	erkenning	vaderschapsactie

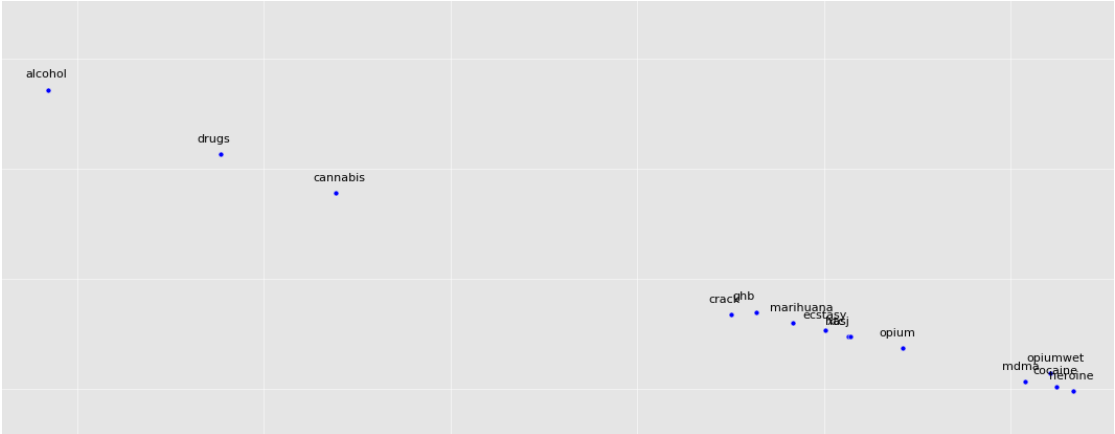
Table 2: A few entries which were found in both the thesauri generated with Word2Vec and the legal thesaurus from the Dutch government. The number of words shown is equal to the number of words in the Dutch legal thesaurus. In general, the related words generated by the model also seem relevant to the topic. Note that some words that are in the ground truth might not be in the vocabulary of the model.



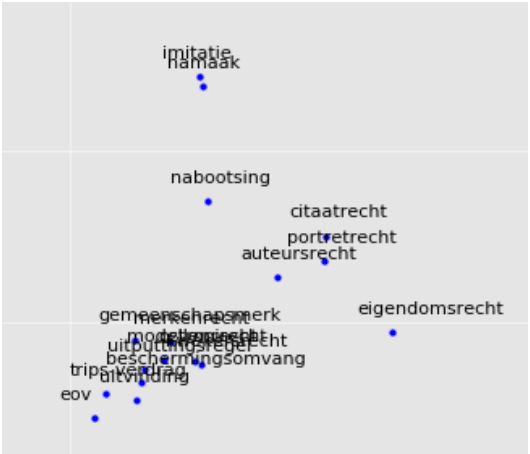
(a) A cluster about vehicles and traffic (laws)



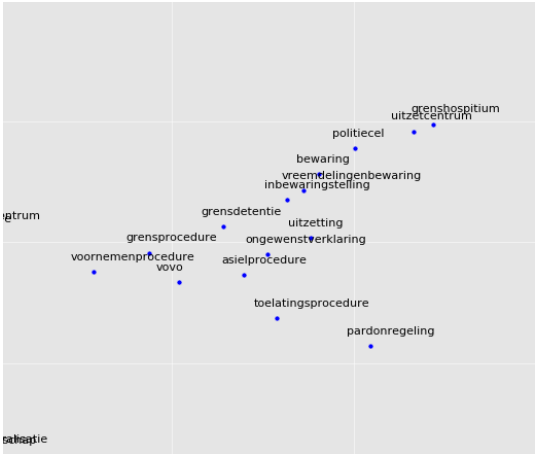
(b) A cluster about education



(c) A cluster about drugs



(d) A cluster about counterfeiting



(e) A cluster about prison and eviction

Figure 3: Different clusters found in the t-SNE embedding of the 300-dimensional vectors produced by Word2Vec. *NB: Zoom in if the text is hard to read.*

as finding one pair of words with a particular semantic relationship (e.g., 'capital of') to find all pairs of words with this relationship (cf. *Paris* − *France* + *Germany* = *Berlin* [1]). The legal thesaurus used for evaluation is already annotated with the semantic relationships, but these annotations were discarded to simplify the task.

The Dutch legal thesaurus might not be the optimal evaluation method. Searching with specialized search engines is generally a recall-centered task. Legal specialists can't afford to miss an important document, so it is better to present more suggestions, even though they may be more noisy. The thesaurus only contains on average 2.5 terms per entry. It might thus be useful to do further research into how a thesaurus is used in practice and what precision/recall trade-offs can be made.

A problem with the approaches presented here is that they have a very large vocabulary. Rather than comparing the vocabulary with the terms in the ground truth thesaurus, it would be useful to reduce the noise in the vocabulary without relying on an already existing thesaurus. A simple approach would be part-of-speech tagging. Since a thesaurus generally only contains nouns, all other words could be filtered. A more advanced approach would be to annotate a subset of the words in the vocabulary with *relevant* or *not relevant*. A classifier could then be trained to find all relevant words.

Lastly, I have not done any experimenting with different hyperparameter settings for both approaches due to time constraints. The current settings were chosen based on suggestions from the documentation of the packages. Optimization of the hyperparameters may yield better results.

5 Conclusion

I have proposed a method for fully unsupervised automatic thesaurus generation for the legal domain using Word2Vec. This approach outperformed a similar method based on LSA. Moreover, it is promising to be able to find semantic relationships between words.

6 References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[2] Xiaobin Li, Stan Szpakowicz, and Stan Matwin. A wordnet-based algorithm for word sense disambiguation. In *IJCAI*, volume 95, pages 1368–1374, 1995.

[3] Gregory Grefenstette. Automatic thesaurus generation from raw text using knowledge-poor techniques. In *Making sense of Words. Ninth Annual Conference of the UW Centre for the New OED and text Research*, 1993.

[4] Gerda Ruge. Automatic detection of thesaurus relations for information retrieval applications. In *Foundations of Computer Science*, pages 499–506. Springer, 1997.

[5] Victoria J Hodge and Jim Austin. Hierarchical word clustering—automatic thesaurus generation. *Neurocomputing*, 48(1):819–846, 2002.

[6] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[7] Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 952–961. Association for Computational Linguistics, 2012.

[8] Antal van den Bosch, Bertjan Busser, Sander Canisius, and Walter Daelemans.

An efficient memory-based morphosyntactic tagger and parser for dutch. *LOT Occasional Series*, 7:191–206, 2007.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[11] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.