Finding Semantically Related Terms on the Web Using Random Indexing: Detecting New Drugs on the Dark Web

Elinor Forsman
Uppsala University Department of Information Technology

November 21, 2015

Abstract

There is an emerging international phenomenon of new drugs that have not previously existed being manufactured and then traded over the Internet. In many cases law enforcement agencies and customs have problems keeping track of all these new drugs causing them to have problem to make appropriate responses to the situation. Because of this measures are now taken to discover these new drugs early as they emerge by developing systems raising warnings when they recognize patterns of new drug trends.

This project investigates ways to find these new drugs as they appear and are being discussed on the Web. Techniques able to interpret the semantic meaning of words, such as the technique random indexing, which has the ability to describe the meaning of words in vectors, are investigated and used to process text gathered from appropriate Internet sources. It is shown that this technique has the capacity to find and extract drug names from text and could be an appropriate component of future systems discovering new drugs being discussed on the Internet.

Acknowledgements

I want to express my gratitude to my supervisor Lisa Kaati and reviewer Michael Ashcroft for all your help, support and guidance throughout this project. I also want to thank Fredrik Johansson and Amendra Shrestha for providing me with all the valuable help, suggestions and comments.

Special thanks to Mark van Staalduinen and Stefan Verbruggen at TNO (Netherlands Organization for Applied Scientific Research) for providing the data sets for the experiments and offering your assistance.

Sincere thanks to Uppsala University and FOI for offering me the devices and resources essential for completing this project.

Contents

Chapter 1

Introduction

Over the recent years there has been an emerging trend of trading narcotics and other psychoactive substances over the Internet, making them increasingly available all over the world. This trend has made it difficult for law enforcement agencies and customs to keep track of all the different names of drugs and medications (including their street names and abbreviations) being manufactured and traded, and thus have problem making appropriate responses to the situation. In some cases available psychoactive substances are newly manufactured compounds that do not fall under present control measures for drug trade, making them legal to use, buy and sell.

The goal of this project is to investigate to what extent automated text analysis methods can be used to identify names of new drugs in written text where drugs are likely to be mentioned.

This is a smaller part of the larger more general problem of finding semantically related terms in texts using automated methods, and will be solved accordingly by using techniques designed to enable computers to interpret the semantic meanings of words.

This master thesis is part of the FP7-project SAFEPOST: "Reuse and Development of Security Knowledge Assets for International Postal Supply Chains". Grant agreement no: 285104. The project was conducted at the Swedish defense research agency, FOI.

1.1 Online Drug Marketplaces

Due to the increasing interconnectedness of the world and the growth of the Internet recent developments have shown that drug markets, including the markets for new drugs, are growing and are predicted to keep growing. Drugs can be manufactured by companies in one country, traded on online drug marketplaces and then shipped to other places in the world. In for example the European Union this has lead to an increase in type, availability and number of drugs and to a growing number of serious harms, and particular challenges have emerged due to the speed of which new drugs appear and the lacking information of their possible harms [1].

Because of the illegal nature of drug trade, many online drug marketplaces exist on the so-called dark Web. The normal Web consists of web pages searchable and indexable by search engines and is the part of the Internet that most people use, and everything that is not searchable and indexable by search engines is refereed to as the deep Web [2], which dark Web is a small portion of. Both the deep Web and the dark Web are considered hidden parts of the Internet, but while deep Web is only hidden because it consists of massive databases that current search engines cannot index efficiently [3], dark Web is deliberately hidden to keep its users and servers anonymous.

1.2 The Onion Router

To reach dark Web it is necessary to use some sort of anonymity software, which provides users and servers with anonymity on the Internet and enables them to establish anonymous connections. One of the most popular anonymity software is called The Onion Router (Tor), which establishes connections between the users and servers while preventing anyone from matching the origin and the destination of traffic sent between them [2]. This is done by encapsulating web traffic between into several layers of encryption (like an onion) and then redirecting this traffic through Tor's volunteer-based anonymity network. The encrypted web traffic travels through a route of randomly selected computers in this network, and at each one of these computers one layer of the encryption is peeled off [2]. It is not until the traffic reaches its destination that all layers have been removed. This way of routing makes sure that no computer in the network both receives traffic from the origin and redirects it to the destination, and thus no computer can match the

origin with the destination.

Alongside this, Tor also provides ability for servers to remain anonymous by allowing them to configure themselves to receive only inbound connections through Tor [2]. Users can access a server through Tor, and Tor routes data between the user and server while hiding the server's IP address. This makes it possible to users to visit the server without the server having to reveal who hosts it [2].

Because of Tor being able to provide anonymity for users and servers it has naturally become a place for illegal online marketplaces to host their servers. Some of the most known online marketplaces are Agora, Silk Road, Black Market Reloaded, and Pandora [4], where Silk Road used to be the most known one, primarily used to sell narcotics and controlled substances [5] before it was shut down by FBI in October 2013 [4]. This marketplace was often compared to eBay because of their similar online infrastructure for customers and vendors, and it includes functionality to conduct online transactions, prohibition of selling harmful products and online chat forums [6]. Data from Silk Road can now be used for research and may be an appropriate resource for discovering drug names.

1.3 Early Warnings

The growth of drug markets is and will continue to be a challenge for public health and drug policies [1]. Legal systems have difficult catching up to the rapid developments of drug trends and online drug trade, that may lead to lack of laws and control measures necessary to keep the situation under control. To respond to this situation, systems are now being developed for raising early warnings when new drugs or drug trends emerge. By raising these warnings the drugs or trends can be further investigated so that appropriate legal and medical responses can be made. Examples of such systems are the so called drug information systems, which collect and analyze information on usage of illicit substances to find emerging trends in drug use [7].

An example of a drug information system is the European Union's Early Warning System, which collects information on new drugs appearing in the European Union, monitor these drugs and raises warning if they appear harmful so that responses can quickly be made [1].

1.4 Problem Formulation and Goals

In this project text will be downloaded from appropriate online marketplaces where new drugs are likely to be discovered and a method called random indexing will be used to interpret the semantic meanings of the words in this text. A list of n known drugs will be used as an input, and words with similar semantic meaning to the words in this list will be considered drug names and returned as result. The purpose of this project is to investigate how well this method works and if it is appropriate for finding new drug names online.

The research questions can be summed up with the following points:

- Is it possible to discover new drug names using automated text analysis methods?
- Is random indexing an appropriate method for discovering new drug names?
- How should the experiments be formed to produce optimal results?

1.5 Delimitations

The text used for discovering drug names in this project is gathered from the dark Web server Silk Road. This choice of source is based on the idea that Silk Road used to be a place where many different kind of drugs were sold and discussed.

This text from Silk Road can then be analysed using different methods and approaches, but in this project only the dimension reduction technique random indexing is considered. Random indexing is chosen because it uses efficient, scalable and incremental methods for interpreting word meanings [8], allowing for text to be analysed quicker, see section 2.2.2 on random indexing.

The goal of this project is to find the name of new drugs, but because of the lack of information on them it would be too difficult to evaluate the results and determine whether new drug names are found or not. Instead the project will investigate how well the method works for finding any drug name, and will use the results as an

indicator of how well the method would work to find new drug names in future systems.

Chapter 2

Background

2.1 Text Mining

Text mining is the process of discovering new or previously unknown information by extracting patterns from written resources and using linguistic or statistical methods to analyse them [9]. Typically machine learning techniques are applied to text mining to help analyse patterns by looking at meaning and relationships in the text's content.

2.1.1 Machine Learning

Machine learning consists of techniques used by computers to adopt and modify their actions so that they become more accurate [10], where the action's accuracy is measured based on how well it can reflect the correct action. If for example the correct action is to win a board game, the accuracy is measured based on often the computer wins the game. Machine learning consists of several different categories such as supervised learning, reinforcement learning and unsupervised learning, where each one learns using different approaches.

The supervised learning technique receives a set of labeled input data consisting several examples pairs of input and correct output [10]. The goal is to come up with a general rule able to predict the desired output based on the input. An example of a supervised learning task is to input a set of images to the algorithm, where some images are of faces and some are not. Each image must be labeled

describing whether there is a face on it. The goal is to come up with a general rule that can predict which pictures have faces on them.

Unsupervised learning is different from supervised learning in the sense that the input data is not labeled and the computer does not know what a correct output is [10]. Instead it tries to see if the inputs have any similarities and groups them. For example can an unsupervised learning algorithm receive input data with images of faces, landscapes and houses where no image is labeled. It can thus not learn how to recognize for example faces, but it can learn that images of faces look very different from images with landscapes and houses.

Reinforcement learning is an algorithm that gets told if an action it has performed is the wrong one, but is not provided any answer on how to correct it [10]. Instead the algorithm has to investigate other actions until it performs the correct one.

The text mining techniques reviewed in this project use unsupervised learning techniques that are given sets of unlabeled input data in the form of written resources.

2.1.2 Written Resource

To perform text mining it is necessary to have a written resource to process and extract information from. These resources can appear in different formats, such as documents or texts represented in hyperlink environments, and are usually gathered from places where it is likely that the desired information is present. When retrieving information about drug trends it is for example appropriate to gather information from sources where drugs are likely to be discussed.

A process called web crawling can be used to collect written resources by downloading the content of web sites [11]. This process usually starts with a list of URL:s, where each one of the URL is accessed and hyperlinks included in it are recursively extracted. This way web crawling follows several paths and retrieve data from web pages as it goes.

2.1.3 Preparing the Written Resources

The text in the written resources are usually prepared before being used for further analysis. It is cleaned from text that does not carry relevant information and key features of the text are identified. This preparation usually includes various steps of processing both individual words and sentences.

HTML Tags

Web pages are written in a tagged mark-up language called HTML, which allows for authors of web pages to for example specify the layout, embed diagrams and create hyperlinks [12]. An example of a mark-up tag that can be used in HTML is a so called anchor tag, which consists of a href attribute pointing to other web pages [12]. An anchor tag could for example look like the following:

$$< a\ href = "http://www.uu.se/" > Uppsala\ University < /a >$$

This href attribute points to the web site of Uppsala University, and it is enclosed within the <a> and < / a> tags.

When written resources are collected from web pages on the Internet, they will initially consist of these HTML tags. Because they are in many cases redundant and do not bring any valuable information for text mining, they are filtered out from the text. They can be filtered by removing tags delimited by the < and > characters only leaving text not enclosed in tags.

An example of a text consisting HTML tags may look like the following:

$$< a\ href = "http://www.uu.se/" > Uppsala\ University < /a >$$
$$< br > Uppsala\ is\ Sweden's\ fourth\ largest\ city\ and\ one\ of\ its\ oldest.$$

After removing HTML tags and the name of the web site the text will look like this:

$$Uppsala\ is\ Sweden's\ fourth\ largest\ city\ and\ one\ of\ its\ oldest.$$

Basic Preparations

Basic preparations of text may include turning upper case letters into lower case letters and removing undesired symbols. This is sometimes necessary to make sure that a word containing upper case letters, like for example words in the beginning of sentences, will be interpreted as the same word as when it occurs with only lower case letters. Undesired symbols that do not carry any necessary information may also be filtered out.

Performing basic preparations on the sentence with removed HTML tags may produce the following result:

uppsala is swedens fourth largest city and one of its oldest

Function Words

In most languages there is a distinction made between words belonging to the lexical category and functional category, where the lexical category includes words such as nouns, verbs and adjectives, and the functional category includes determiners, prepositions, auxiliaries, conjunctions and other sorts of particles [13]. Examples of English function words are *"a"*, *"the"* and *"on"*.

While lexical words tend to occur in contexts related to their meaning, function words occur often in all kinds of contexts and tend to be carrying little meaning. The performance of text mining may be improve by removing these function words since they will usually not carry meaning and without them there will be less words to process [12]. Removing the function words from the sentence without HTML tags and with basic processing will leave a sentence only consisting of the lexical words:

uppsala swedens fourth largest city one oldest

Lemmatization

Due to grammatical reasons the words of a language often occur in different forms. For example are the words *"organize"*, *"organizes"* and *"organizing"* all related to

each other and carry similar meaning, but are spelled differently and may be used in different ways. Lemmatization is a technique aiming to reduce these types of inflectional forms of similar words into one and the same form, usually their base or dictionary form [14]. This is done because even though some words are spelled differently they still carry similar meaning, something that a computer cannot easily recognize. Even though some information will be lost when doing this, it can be a better alternative then not having the computer understand that they are related at all. The process of lemmatization includes vocabulary and morphological analysis of words to reduce their endings if they are inflectional.

Doing lemmatization on the sentence without HTML tags, with basic processing and without function words could give the following result:

$$uppsala\ swedens\ fourth\ large\ city\ one\ old$$

It can be discussed whether the word *"fourth"* should have been lemmatized into *"four"* or not. These types of issues makes lemmatization tasks difficult causing lemmatization tools to be incomplete.

Stemming

Stemming is a technique similar to lemmatization, aiming to reduce inflectional forms of similar words but with a different approach. Instead of reducing the words into their base or dictionary forms, stemming chops off the ending of words with the hope of getting this correctly most of the time [14]. For example would the verb *"saw"* be lemmatized into *"see"* but stemmed into *"s"*.

Part of Speech Tagging

In languages words are grouped into categories of syntactic or grammatical nature, where the most basic ones are nouns, adjectives and verbs. These categories are called part of speech (POS) and are defined in various dictionaries and lexicons [11]. In text mining a so called POS-tagger can be used to label words with the part of speech category that they belong to. If a word belongs to multiple part of speech categories a triggering condition can be used to determine which POS-tag the word should be labeled with. This can for example be done by looking at POS-tags of closely occurring words.

If for example a word belongs to both the noun and verb categories, a triggering condition can state that it will be labeled as a verb if it occurs after the word "to". If it does not occur after the word "to" it is labeled being a noun. Lets say we have the following sentence:

he needs to map the region

In this sentence the word *"map"* belongs to multiple categories, namely noun and verb. Initially it is POS-tagged as a noun, and the word occurring in front of it is investigated. This word does is indeed *"to"* and the triggering condition is met. The word *"map"* will now instead be labeled being a verb.

POS-tagging can be used in text mining when the part of speech categories are relevant information. Performing POS-tagging on the sentence above will show that the word *"he"* is a personal pronoun, *"needs"* and *"map"* are verbs, *"to"* has a to-label, *"the"* is a determiner and *"region"* is a noun.

Note that words can get a different POS-tag before and after they have been lemmatized. For example is the word *"different"* POS-tagged as an adjective, but if it is lemmatized into *"differ"* it is instead POS-tagged as a verb. In this project this issue is however not dealt with since POS-tagging will not be used in the initial text processing step and storing information about each word's POS-label from before they are lemmatized would be insufficient. Also only drug names will be POS-tagged and it will be assumed that most of their POS-labels will not change due to lemmatization.

2.2 Semantically Related Terms in Text

Once the text is prepared and ready for text mining it is time to choose methods appropriate to discover information in it. This project will focus on methods belonging to so called word space models, which are techniques using distributional statistics to generate vectors for words in a text. These vectors make up a high-dimensional vector space and their directions can be compared to find semantic similarity between the words they represent [16].

2.2.1 Basics of Word Space Models

Word space models can capture the semantic meaning of words by keeping statistics about them in vectors or matrices [8]. The statistics alone do however not provide us with much information, but it is rather how the words' vectors relate to each other that can be used to find knowledge about the words [17]. Word space models can thus be used to discover semantic relatedness between words, which can be used to solve linguistic tasks.

Word-Context Matrices and Context Vectors

The vectors storing statistics about words can be refereed to as context vectors while the matrices can be refereed to as word-by-context matrices. Each row in a word-by-context matrix describes an unique word, and context vectors can be generated from this matrix by extracting each row from it [8]. Because of this a word-by-context matrix can be seen as a set of many context vectors.

$$Matrix = \begin{bmatrix} context\_vector_1 \\ context\_vector_2 \\ context\_vector_3 \\ context\_vector_4 \end{bmatrix}$$

The statistics for every word in the matrix is gathered by looking at what contexts each word occurs in and keeping count of how often they occur in each context. The columns of the word-by-context matrix specify these contexts. What a context is may differ and could for example be documents in which the words occur in [8]. In this case the columns of the matrix represent the documents and the matrix is called a word-by-document matrix [8]. A context could also be other words co-occurring with the word of the row. Each column of this matrix then represents each possible word that could co-occur, and the matrix is in this case called a word-by-word matrix [8]. The following matrix specifies a word-by-context matrix where it is kept count how often $word_1$, $word_2$, $word_3$ and $word_4$ occur in the contexts $context_1$, $context_2$, $context_3$ and $context_4$. For example is $word_2$ occurring 0 times in $context_1$, 1 time in $context_2$, 0 times in $context_3$ and 2 times in $context_4$.

$$Matrix = \begin{bmatrix} & context_1 & context_2 & context_3 & context_4 \\ word_1 & 3 & 0 & 6 & 1 \\ word_2 & 0 & 1 & 0 & 2 \\ word_3 & 1 & 1 & 3 & 0 \\ word_4 & 0 & 2 & 2 & 2 \end{bmatrix}$$

The following vectors describe the context vectors of $word_1$, $word_2$, $word_3$ and $word_4$ extracted from previous matrix.

$$
\begin{aligned}
word_1 &= \begin{bmatrix} 3 & 0 & 6 & 1 \end{bmatrix} \\
word_2 &= \begin{bmatrix} 0 & 1 & 0 & 2 \end{bmatrix} \\
word_3 &= \begin{bmatrix} 1 & 1 & 3 & 0 \end{bmatrix} \\
word_4 &= \begin{bmatrix} 0 & 2 & 2 & 2 \end{bmatrix}
\end{aligned}
$$

Word-By-Word Matrices and Window Size

A word-by-word matrix is a version of the word-by-context matrix where the context of each word are other words it is co-occurring with [8]. If for example $word_2$, $word_3$ and $word_4$ occur close to $word_1$ they are considered to be the context of $word_1$ and their co-occurrences will be stored in $word_1$'s row in the word-by-word matrix. A word-by-word matrix could look like the following:

$$
Matrix = \begin{bmatrix}
 & word_1 & word_2 & word_3 & word_4 \\
word_1 & 0 & 1 & 1 & 1 \\
word_2 & 1 & 0 & 2 & 0 \\
word_3 & 1 & 3 & 0 & 3 \\
word_4 & 1 & 0 & 2 & 0
\end{bmatrix}
$$

How close $word_2$, $word_3$ and $word_4$ need to be to $word_1$ in a text to be considered $word_1$'s context is determined by a window size.

Say that we have the following sentence:

the cat and the dog

If the window size is set to 1 and a context vector for the word *"and"* is generated, words occurring one step before and one step after *"and"* will then be considered context, in this case *"cat"* and *"the"*. If for example *"the"* would occur several times close to *"and"*, each one of these instances is counted.

If instead the window size is set to 2, words occurring two steps before and two steps after *"and"* are considered context, in this case *"the"*, *"cat"* and *"dog"*. The

word-by-word matrix of this sentence using the window size 2 would look like the following:

$$Matrix = \begin{bmatrix} & the & cat & and & dog \\ the & 0 & 2 & 2 & 1 \\ cat & 2 & 0 & 1 & 0 \\ and & 2 & 1 & 0 & 1 \\ dog & 1 & 0 & 1 & 0 \end{bmatrix}$$

The Distributional Hypothesis

Because context vectors are built by looking at what contexts their words occur in, it can be said that words that have occurred in similar contexts will have similar context vectors, and words with similar context vectors have occurred in similar contexts, see equation 2.1.

$$Words\ occur\ in\ similar\ contexts \iff Words\ have\ similar\ context\ vectors \tag{2.1}$$

Many word space models are based on the Distributional Hypothesis, which states that *"words with similar meaning tend to occur in similar contexts"* [8]. In this project it will be assumed that this hypothesis holds true. Based on this assumption, and because words occurring in similar contexts will have similar context vectors, it can be assumed that words with similar meaning will tend to have similar context vectors. The opposite is also assumed to hold true, so words with similar context vectors will tend to have similar meaning, see equation 2.2.

$$Words\ have\ similar\ meaning \iff Words\ have\ similar\ context\ vectors \tag{2.2}$$

If words are considered being contexts, this means that two words usually surrounded by the same sorts of words will be similar in meaning.

High-Dimensionality Spaces

By using the word-by-context matrices and context vectors, word space models are able to associate the words in natural language texts with the contexts that they occur in. Each word has a context vector of their own that can be represented in a high-dimensional vector space, and the directions of context vectors can be used to indicate semantic similarity between them [8].

Because the dimensionality of word-by-context matrices, context vectors and vector spaces increase with the number of possible contexts, it will grow very large very quickly when there are many contexts present. If for example words are considered being context, the dimensionality will be as large as the vocabulary [8]. These large sizes will make matrices to be difficult to compute, and because only a small amount of the possible co-occurrences in the matrices will actually occur, most cells in them will be zero [8]. This problem is referred to as the high-dimensionality problem and is an issue that word space models have to deal with.

2.2.2 Word Space Model Techniques

There exist various different word space model techniques and approaches, but this project will focus on only describing two techniques called latent semantic analysis and random indexing.

Latent Semantic Analysis

Latent semantic analysis (LSA) is an unsupervised learning technique only requiring a large set of texts as input, and attempts to find similarity structures in these texts by looking at what contexts words occur in [18]. If documents are considered contexts, the technique starts off with building a word-by-document matrix containing statistics on how frequently words occur in each document, and it could for example look like the following:

$$Matrix = \begin{bmatrix} & document_1 & document_2 \\ word_1 & 1 & 0 \\ word_2 & 0 & 1 \\ word_3 & 2 & 2 \\ word_4 & 0 & 2 \end{bmatrix}$$

The word frequencies of each cell are then usually transformed by for example cumulating them inversely based on the total number of word occurrences in the data set [18] using the method inverse document frequency (IDF). This method weights the number of documents a word has occurred in with the number of documents in the whole data set. This is done because some words bring more meaning to their contexts than others and all words should not all be considered equally important [20]. It is assumed that words occurring in a few set of contexts

tend to better capture the meaning of those contexts than words occurring in many contexts [19]. For example could the word *"botcott"* occur in a few documents while the word *"somewhat"* occur in many, and if this is the case, *"botcott"* will probably better capture the meaning of its documents and should thus be seen as a better keyword. IDF gives *"botcott"* more weight than *"somewhat"* by comparing the number of documents each of them has occurred in with the number of documents in the whole data set using equation 2.3, where w is the word, D is the number of documents in the set and df_w is the number of documents containing the word w.

$$idf(w, D) = -log_2(\frac{df_w}{D}) \tag{2.3}$$

If the input data is very large and contains many words and documents it is likely that the produced word-by-document matrix is high in dimensionality, causing the high-dimensionality problem 2.2.1. LSA attempts to solve this problem by using dimension reduction techniques such as singular value decomposition (SVD). SVD divides the word-by-document matrix up into the three matrices T, S and D, where T consist of term vectors, D consists of document vectors (both used for information retrieval problems) [18], and S is a diagonal matrix containing entries sorted in decreasing order. LSA uses a reduced-dimension SVD where the k largest entries in the S matrix are used including their associated term and document vectors in matrix T and D, resulting in a reduced dimensional approximation of the original word-by-document matrix with only k parameters [18]. When all this is done each row of the processed word-by-document matrix can be extracted and used as context vectors.

Random Indexing

Random indexing is a word space model technique that attempts to extract the meaning of words by producing context vectors for words using either documents or co-occurring words as context [8]. This is the word space model that will be used for this project, and for it co-occurring words will be considered context. Random indexing attempts to solve the high-dimensionality problem by giving context vectors significantly lower dimensionality than the number of contexts in the expression [8]. Instead of gathering statistics about co-occurrences in a word-by-context matrix, each word is assigned a so called random index vector (which can be seen as an identification for the word it is assigned to) that can be used as an alternative to capture these co-occurrences.

The first step of the random indexing algorithm is to generate these random index vectors with a random distribution of +1s, -1s and 0s and then assign them to

each word [8]. How long these vectors are still depend on the number of contexts in the text, but it is not necessary to introduce one new dimension for every new context. Instead it is only necessary to have a dimensionality only need to be big enough for each word to get their own unique random index vector, which usually will be significantly lower than the number of contexts. Using for example three dimensions random vectors for the following six words can be generated:

Context Vectors
$word_1$: (0 1 -1)
$word_2$: (0 -1 1)
$word_3$: (1 0 -1)
$word_4$: (1 -1 0)
$word_5$: (-1 0 1)
$word_6$: (-1 1 0)

The dimensionality of random index vectors are set before statistics about word co-occurrences are gathered and does not grow after this point [8].

In the next step every word in the text is assigned with an empty context vector, which contains only zeros and has the same dimensionality as the random index vectors. These context vectors will be filled with statistics about word co-occurrences by having the random index vectors of co-occurring words be added to them. This way the context vectors will contain a distribution of numbers that describe which words they have co-occurred with. Random indexing successfully solves the high dimensionality problem by making dimensionality depend on the size of the random index vectors, which is significantly smaller than the vocabulary of the text.

In the following example the words $word_1$, $word_2$, $word_3$ and $word_4$ are first assigned an empty context vector and a random index vector each:

Context Vectors
$word_1$: (0 0 0)
$word_2$: (0 0 0)
$word_3$: (0 0 0)
$word_4$: (0 0 0)

Random Index Vectors
$word_1$: (1 0 -1)
$word_2$: (0 1 -1)
$word_3$: (0 -1 1)
$word_4$: (-1 1 0)

If for example $word_1$ co-occurs with $word_2$ and $word_4$ in the text, the random index vectors of both $word_2$ and $word_4$ are added to the context vector of $word_1$. The updated context vectors will then look like this:

Updated Context Vectors
$word_1$: (-1 2 -1)
$word_2$: (0 0 0)
$word_3$: (0 0 0)
$word_4$: (0 0 0)

Say that we have the following sentence:

study computer science in Uppsala

If the window size is set to 2 and each word's random index vector looks like the following:

Random Index Vectors
study : (1, 1, 1, 0, -1)
computer: (0, 0, 0, -1, 0)
science : (-1, 0, 0, 1, 0)
in : (0, 0, 1, 1, 0)
Uppsala : (0, 0, -1, 1, 1)

The random index vectors of the words occurring two steps before and two steps after each word will be added to their context vector. If we look at the word *"science"* in the example sentence, the random index vectors of the words *"study"*, *"computer"*, *"in"* and *"Uppsala"* will all be added to *"science"*'s context vector, see Figure 2.1, and will look like the following:

science : (1, 1, 1, 1, 0)

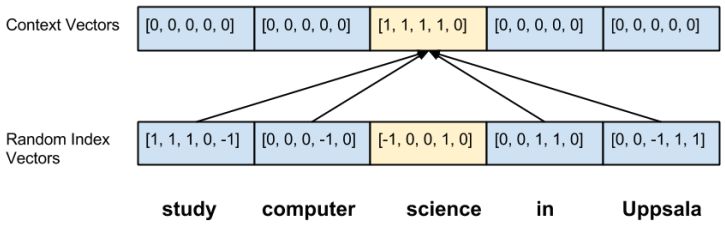

Figure 2.1: Random index vectors of *"study"*, *"computer"*, *"in"* and *"Uppsala"* being added to *"science"*'s context vector.

Because the fixed dimensionality of context vectors is significantly lower than the number of contexts, random indexing has solved the high dimensionality problem and there is no need to apply expensive dimensionality reduction algorithms after when statistics about word co-occurrences have been gathered, which leads to gains in processing time and memory consumption [8].

Word2Vec

Word2Vec is another technique representing words with vectors to describe their semantic meaning and to use for carrying out natural language processing tasks [21]. The technique first generates word pairs (w, c) from the input text [21] for words w and their contexts c (which are words occurring within a window around w [22]). It tries to maximize the probability of words co-occurring having similar vectors and minimize the probability of words not co-occurring having similar vectors.

The technique uses the two parameter update equations skip-gram and continuous bag of words (CBOW) to maximize the probability of word vectors v_w being similar to context vectors v_c for each word pair, which is done by in steps of moving v_w and v_c closer to or farther away from each other [21]. Essentially skip-gram and CBOW are opposites of each other, where skip-gram takes a word as input to find its contexts [22] while CBOW takes the contexts as input to find the word [21].

A mathematical explanation for the process of maximizing the probability of similar word having similar vectors can be seen in equation 2.4 and 2.5, which are used in the skip-gram model. Equation 2.4 is used to maximize the overall probability

for all word pairs D in the text having similar vectors, where the θ parameters v_w and v_c are set to maximize this probability. Equation 2.5 is used to calculate the probability of a word vector v_w being similar to its context vectors v_c, where C is the set of available contexts.

$$arg \max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta) \tag{2.4}$$

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}} \tag{2.5}$$

The problem with this approach is that it can be expensive to compute the overall probability for all contexts in $\sum_{c' \in C} e^{v_{c'} \cdot v_w}$ of equation 2.5, because it is too many word pairs to calculate each iteration [22]. This can be solved using for example negative sampling, which samples only some contexts to compute the words pairs scores for instead of all of them [21].

2.2.3 Similarity Measures

The statistics gathered in context vectors can now be used to discover semantic relatedness between words. This is done by comparing the length and direction of the context vectors to see how similar they are, and if it is found that they are similar it is assumed that their words are semantically related (based on the Distributional Hypothesis, see section 2.2.1).

Similarities between vectors can be computed using either similarity measures, which provides high scores for similar objects, or distance measures, which provide low score for the same objects [17].

Dot Product

A simple similarity measure is the so called dot product, which measures the similarity between a vector p and q by calculating their scalar product [17], see equation 2.6.

$$similarity(p,q) = p \cdot q = p_1 q_1 + p_2 q_2 + ... + p_n q_n \tag{2.6}$$

Euclidean Distance

Euclidean distance is a measure calculating the distance between the vectors p and q using equation 2.7 [17].

$$distance(p, q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2.7}$$

Manhattan Distance

The so called Manhattan distance, measures the distance between to vectors p and q by summing up the absolute differences of their Cartesian coordinates [23]. The Manhattan distance is shown in equation 2.8:

$$distance(p, q) = ||p - q|| = \sum_{i=1}^{n} |p_i - q_i| \tag{2.8}$$

Cosine Similarity

Neither the dot product, Euclidean distance or Manhattan distances are however ideal to use for word space models. This is because dot product tends to measure frequent words being too similar to other words [17], while the Euclidean and Manhattan distances will measure frequent words being too far from other words [17]. These problems can be solved if the effects of the vector's magnitude is factored out by normalizing the vector length, see equation 2.9. Because it is the direction of vectors that indicate semantic similarity [8], normalizing the vector length will leave a vector usable when searching for semantically similar terms in text.

$$|p| = \sqrt{p \cdot p} \tag{2.9}$$

Cosine similarity is a measure using this method of normalizing vector length and then calculates the cosine of the angle between the normalized vectors [17], see figure 2.2, in a n-dimensional vector space using equation 2.10.

$$similarity(p, q) = cos(\theta) = \frac{p \cdot q}{||p||\ ||q||} = \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n}(p_i)^2} \times \sqrt{\sum_{i=1}^{n}(q_i)^2}} \tag{2.10}$$

This way the directions of the vectors are compared and if they point in similar directions the similarity measure will be high.

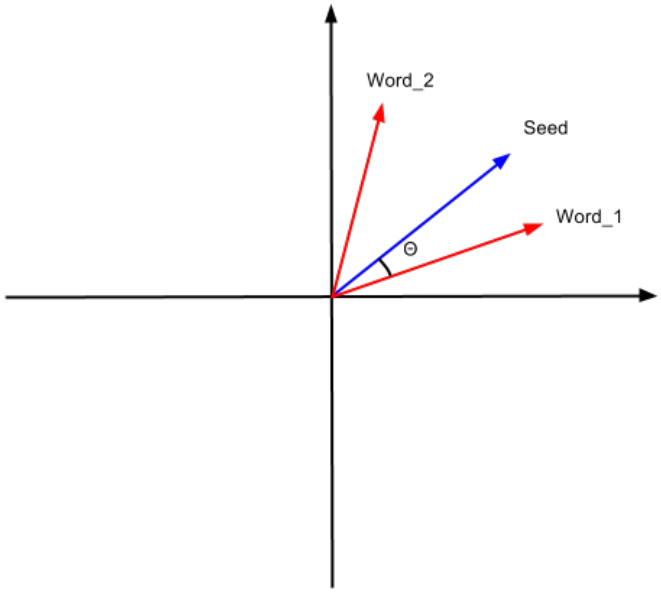

Figure 2.2: A 2-dimensional representation of a vector space containing context vectors. The similarity of the context vector for *"seed"* and $word_1$ is determined by the angle between them.

Seeds and Extraction Criteria

If we now have a set of context vectors for every word in a text and we want to extract words of a specific semantic meaning, a way to do this is to use so called seeds. Seeds are words known to be semantically related to the words being looked for. If we for example want to find words such as *"big"*, *"massive"*, the word *"large"* can be used as an appropriate seed. The seed's context vector is first extracted (assuming it is present in the set of context vectors) and is then compared to all the other context vectors using a similarity function. Words with similar context vectors are returned as result, and if the algorithm worked well the result should contain words semantically related to *"large"*.

How similar words need to be for them to be extracted as results is defined by a so called extraction criteria. An extraction criteria defines a threshold, where for example only words with similarity measure above a certain value are extracted. The extraction criteria can also define how many words should be extracted, and

if this number for example is set to 100, the 100 most similar words are returned as result.

2.2.4 Evaluating the Results

The words now extracted and returned as result are predicted to be semantically similar to the seeds, while the words not extracted are instead predicted to not be semantically similar to the seeds.

To evaluate these results we need to be known which words are actually semantically similar, and which words not. This can be done using either automated methods, which can for example compare the extracted words with a list of words known to be semantically similar, or human expertise, where a person verifies the extracted words' and seeds' similarities.

In this evaluation process each word in the results is labeled as true positives (tp), false positives (fp), false negatives (fn) or true negatives (tn). A true positive is a word that is predicted to be similar and actually is similar, a false positive is a word that is predicted to be similar but actually is not, a false negative is a word predicted to not be similar but actually is similar and a true negative is a word predicted to not be similar and actually isn't, see table 2.1.

	Predicted similar	**Predicted not similar**
Actually similar	True positive	False negative
Actually not similar	False positive	True negative

Table 2.1: Confusion matrix.

When all words have been labeled the performance of the algorithm can be measured by calculating for example accuracy, which is a measurement calculating how close a result is to the reality [24]. The accuracy gets good if there are many semantically similar words among the extracted and not many semantically similar words among the words not extracted. Accuracy is calculated using equation 2.11.

$$Accuracy = \frac{tn \ + \ tp}{tn \ + \ fp \ + \ fn \ + \ tp} \tag{2.11}$$

Measuring the accuracy can however give misleading measurements of quality, and in some cases algorithms with higher predictive power get lower accuracy measures than algorithms with lower predictive power. If for example one algorithm predicts 9700 true negatives, 100 true positives, 150 false positives and 50 false negatives, the accuracy is calculated to be 98.0%, and an other algorithm predicts 9850 true negatives, 0 true positives, 0 false positives and 150 false negatives, the accuracy is calculated to be 98.5%. The second algorithm gets a higher accuracy measure, but does not find any true positives making it have worse predictive power than the first algorithm. This phenomenon is called the accuracy paradox [25].

$$First\ algorithm = \frac{9700 + 100}{9700 + 150 + 50 + 100} = 98.0\%$$
$$Second\ algorithm = \frac{9850 + 0}{9850 + 0 + 150 + 0} = 98.5\%$$

Precision and recall are two other performance measurements that can be used as alternatives to accuracy. Precision is the estimated percentage of how many words in the result that are relevant [11], so how many of the extracted words that are indeed semantically similar to the seeds. The measurement divides the true positives with all the extracted words (true positives and false positives), see equation 2.12.

$$Precision = \frac{tp}{tp\ +\ fp} \tag{2.12}$$

Recall is the estimated percentage of how many of the relevant words in the input text that are present in the result [11], so how many of the words that are semantically similar to the seeds present in the input text that are present among the extracted words. This measurement divides all true positives with all words in the input text that are actually semantically related (true positives and false negatives), see equation 2.13.

$$Recall = \frac{tp}{tp +\ fn} \tag{2.13}$$

It is possible for an algorithm to get a good precision but a bad recall and vice versa. If for example the result contains many true positives but also many false negatives, the precision of the algorithm is good while the recall is bad. And if the result contains only a few false negatives but many false positives, the recall of the algorithm gets good while the precision gets bad. Because of this the precision and recall can be combined to show the trade-off between them so that an algorithm with both good precision and recall can be built.

A method used to combine precision and recall is the so called the precision-recall break even point technique. It balances the precision with the recall by returning the best result when they are equal to each other [11].

An other method combining precision and recall is the F-score technique, which weights precision with recall and has the ability to put more emphasis on one or the other [11], see equation 2.14. If for example $\beta = 2$ the recall weights more than the precision, and if $\beta = 0.5$ the precision weights more than the recall.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \ \cdot \ recall}{(\beta^2 \cdot precision) \ + \ recall} \tag{2.14}$$

2.3 Related Work

The purpose of this project is to develop an algorithm able to find drug names in an input text. This task will be solved by inserting a list of known drug names used as seeds into the algorithm, which then extracts words semantically similar to them. These semantically similar words are assumed to be drug names. Similar research has been done where words are inserted as seeds into an algorithm that finds synonyms or other semantically related words.

2.3.1 Vector-Based Semantic Analysis: Representing Word Meanings Based on Random Labels

In the article [16], Sahlgren presents a method for extracting synonyms from an input text using random indexing. Vectors are produced for each term in the text containing their co-occurrences, and because this method is based on the Distributional Hypothesis, these vectors describe the words' meaning.

The article starts of by discussing the philosophical motivation of using vectors to represent the meanings of words. It is said that the meaning of a word is known if the word can be used correctly in linguistic praxis. This way, if a vector containing statistical information about a word's co-occurrences can be used to carry out linguistic tasks, this vector has successfully captured the meaning of the word. It is however not enough to just assign these vectors to words to describe their meaning; it is only in relation to each other that these vectors really mean

anything. By comparing the directions of vectors conclusions can be made about their words' semantic similarities.

Experiments are made where the input text is first processed in seven different ways to investigate which processing produces the best results. In four of these processings the words in the text are shortened into a number of 6, 8, 10 or 12 characters, and if for example words are shortened into 6 characters, words such as *"disembark"* and *"disembarking"* are both represented with the 6 letter long approximation *"disemb"*. In two of the processings words are stemmed, and in one of these POS-tagging is used. Lastly one experiment is conducted with no processing of the text at all. For all these different processings the context window is varied in sizes between 1, 2, 3 and 4, resulting in a total of 28 experiments that are compared to find which one produces the optimal result. Once the text is processed the random indexing algorithm is used to generate context vectors for the words in the text. Seeds are inserted and the most similar words to them extracted. The result is evaluated against the standard TOEFL vocabulary test to find if indeed synonyms are extracted.

It is found that stemming produces the best result out of all the text processings, and it performs especially well when the window size is set to 3 where 72% of the extracted words are synonyms. This may indicate that using stemming to analyse inherent structural relations and a small window size to discover distributional patterns is important when retrieving semantic information from text [16]. The best overall results for window sizes were found when using the window sizes 2 and 3.

These experiments show that using random labeling of words and small window sizes are appropriate techniques for producing context vectors and can be used for finding synonyms. The question is raised whether the representations of words would get better if for example syntactical information about sentence structure and syntax are also incorporated into the context vectors. Doing this it can be investigated how grammatical information may affect words' semantic representations.

2.3.2 Testing Semantic Similarity Measures for Extracting Synonyms from a Corpus

In the article [26], the author Ferret describes a method for extracting synonyms from text using a corpus-based approach based on the Distributional Hypothesis. The AQUAINT-2 corpus is a text containing about 380 million words gathered from different news articles, and is used as input and distributional data is gathered from it. As preparation the text is lemmatized and all words but nouns, verbs and adjectives are removed.

Co-occurrences for each term of the corpus are then collected in context vectors and the cosine similarity is applied to find the N most similar words to seeds. Once the N words are extracted they are evaluated against the WordNet-based Synonymy Test and the Moby thesaurus, which contain information about synonyms and similar words for nouns [26], to see if the extracted words are indeed synonyms.

The results show that the ability to catch semantic relatedness of seeds seems to be correlated to how frequently they occur in the text. High frequency words produce the best results, while words occurring less than 100 times perform significantly worse. It is also shown that the results get better when the resource used for evaluation contain many synonyms and similar words. However there are issues of evaluating the results because of a lack of consensus on what relations between words should be found, making it difficult to compare the results of these experiments with other similar works.

2.3.3 Commercial products

There also exist commercial products using similar techniques to processing and analysing texts. Gavagai is a company that develops techniques that learn the meaning of words by relating them to their usages and contexts. One of their products is the is the *"Gavagai Living Lexicon"* that is trained on documents from social and news media to model the meaning of words and gather knowledge about them [27].

Chapter 3

Experimental Setup

For this project textual data is gathered from online drug marketplaces where new drugs are likely to be mentioned. The random indexing algorithm is used to gather statistics about words in this data, and cosine similarity measures are used to identify similarities between the words. A N-word long list with known drug names is used as input to this method and the on average the M most similar words to all these N drug names are extracted and represented in a M-word long result list.

3.1 Data Sets

The Netherlands Organization for Applied Scientific Research (TNO) provided this project with data gathered from the now closed dark Web server Silk Road containing text and various discussions made on drug related topics. TNO is a research organization active in the Netherlands that has collected a large set of data from the dark Web, including data from marketplaces such as Agora, Silk Road, Evolution, Black Market Reloaded, and The Pirate Market.

Two data sets were provided, containing 500 MB and 1,25 GB data with posts made on Silk Road, and they were used to develop the techniques and to conduct the experiments.

3.2 Methodology

The experiments consist of preparing the input text, extracting drug names, after processing the extracted drug names and evaluating the result to determine their performance, a graphic representation of the process is shown in figure 3.1.

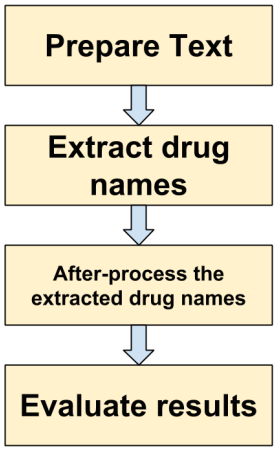

Figure 3.1: Experiment steps.

3.2.1 Preparing the Text

The preparation of the text include removing HTML tags, quotes, user names and hyperlinks, as well as basic processing of making upper case letters to lower case letters, removing non-letter symbols and filtering out function words.

3.2.2 Extracting Drug Names

Context vectors for all words in the input text are produced using random indexing. A list containing 623 known drug names is then prepared by removing all non-letter characters, making upper case letters to lower case letters, shuffling the drug names and finally extracting N of them to be used as seeds. The seeds' context vectors are then extracted and compared to the rest of the vocabulary's context vectors using the cosine similarity measure. The M most similar words are returned as results and represented in a M-word long result list.

3.2.3 After-Processing

When the most similar words are being extracted, they are after-processed before they are entered into the M-word long result list. This after-processing includes removing known drug related word such as *"smoke"*. This is done to filter out words that may get a high similarity measures but are probably not drug names. The M most similar words that are not filtered out in this process are returned as result.

3.2.4 Evaluating Results

The words in the result list are evaluated to see how well the algorithm worked. Each word in the result list is manually evaluated and labeled as a true or a false positive, and lastly the precision is calculated.

The evaluation will consider how many drug names are present in the result list, and not just how many *new* drug names there are. This is because it would be very difficult to evaluate which drug names are new or not, and evaluating how many known drug names the algorithm can catch also gives an idea of how relevant it would be to use in a system searching for new drug names.

In some cases it is not obvious if the words in the result lists should be labeled as drug names or not. To deal with this issue a couple of rules are established on when to label a word as a drug name if it is not already clear. These rules look like the following:

- Multiple drug names being written as one. For example the word *"clonazepamflubromazepam"*, which consists of the two drug names *"clonazepam"* and *"flubromazepam"*.

- Multiple words being written as one where one word is a drug name. For example the word *"mdmamg"*, which consists of the two words *"mdma"* and *"mg"*, where *"mdma"* is a drug name.

- Drug names without prescription. For example *"voltaren"*, which is a prescription free pain killer.

- Misspelled drug names. For example is the word *"exctasy"*, which is a misspelled version of *"ecstasy"*.

These rules are necessary because there only exist two possible labels for words in the result list, namely true positive or false positive, where neither one of them seem like good fits. It is decided that labeling these words as true positives better reflect their semantic meaning than labeling them as false positives.

The results will be evaluated calculating only the precision. The reason for this is due to there not being any information on how many drug names are present in the text, making it difficult to estimate number of false negatives and to calculate recall.

It can also be argued that precision is also more interesting and relevant information for this project than recall is. A high precision shows that the words the algorithm finds are indeed drug names, and because this algorithm should be appropriate for an early warning system raising warnings when new drug names emerge, a high precision makes sure that warnings are not raised (so often) unless indeed drug names are found. Having a high recall makes sure that most drug names in the text are detected, but does not promise that the detected words are mostly drug names. This may not work very well for an early warning system since it could result in many warnings being raised even though drug names have not been found.

3.3 Text Processing Methods

For the experiments three different text processing methods developed and tested to see how different preparations of text and after-processing affect the result. Each one of these processing methods follow the same process of preparing the input text, extracting drug names, after-processing and evaluation, but with slightly different approaches.

3.3.1 Standard

The first method uses a standard processing following the method described above. It removes HTML-tags, citations, quotations, non-letters and function words, as well as make upper case letters to lower case. In the after-processing step, drug related words are removed.

3.3.2 Lemmatization

The second method is called *"Lemmatization"*, which uses the same approach as the standard processing with an added feature in the preparation step. Once all the basic preparations are made all words in the remaining text are lemmatized. This is done to make words with inflectional endings bring the same meaning to their contexts.

3.3.3 StandardPOSFilter

The third method is called *"StandardPOSFilter"*, which uses the same approach as the standard processing with an added feature in the after-processing step. It uses a POS-tagger which labels the extracted words with the POS-category they belong to. All words but nouns are then filtered out, leaving only nouns in the result list. This feature is added based on the idea that drugs names will mostly be labeled as nouns.

3.4 Experiments

Three different experiments are conducted to evaluate how the results differ depending on the processing method used, number of seeds inserted and how many words are extracted as result.

3.4.1 Experiment 1: Examining Text Processing Methods

In the first experiment the three different text processing methods are evaluated. Around 207 known drug names are used as seeds and the 100 most similar words are extracted as result. Each text processing method is tested 3 times with 3 different seeds list (all containing around 207 known drug names) and the average precision of each is calculated.

3.4.2 Experiment 2: Examining Number of Seeds

The second experiment investigates how the result varies depending on the size of the N-word input list of seeds. The size of this list varies in steps of 100, 200, 300, 400, 500 and 600, and for each one of these the 100 most similar words are extracted and the precision calculated.

3.4.3 Experiment 3: Examining Extraction Criteria

For the last experiment it is investigated how the number of similar words extracted affects the result. The number extracted words varies in steps of 100, 200 and 300, and the number of seeds that provided the best results in experiment 2 will be used as input. The result is evaluated by calculating the precision.

Chapter 4

Experimental Results

4.1 Experiment 1: Examining Text Processing Methods

In total 9 different experiments are conducted in this part. The seed list is divided up into three smaller lists of seeds containing about 207 known drug names each, and are all used for each one of the three text processing methods. The number of drug names found in each experiment is shown in table 4.1, and the precision for each method is calculated based on their average result and shown in figure 4.1.

	First seed list	Second seed list	Third seed list	Average
Standard	69	78	79	75.33
Lemmatized	63	81	79	74.33
StandardPOSFilter	73	85	86	81.33

Table 4.1: Results for experiment 1 showing the number of drug names found for each text processing method.

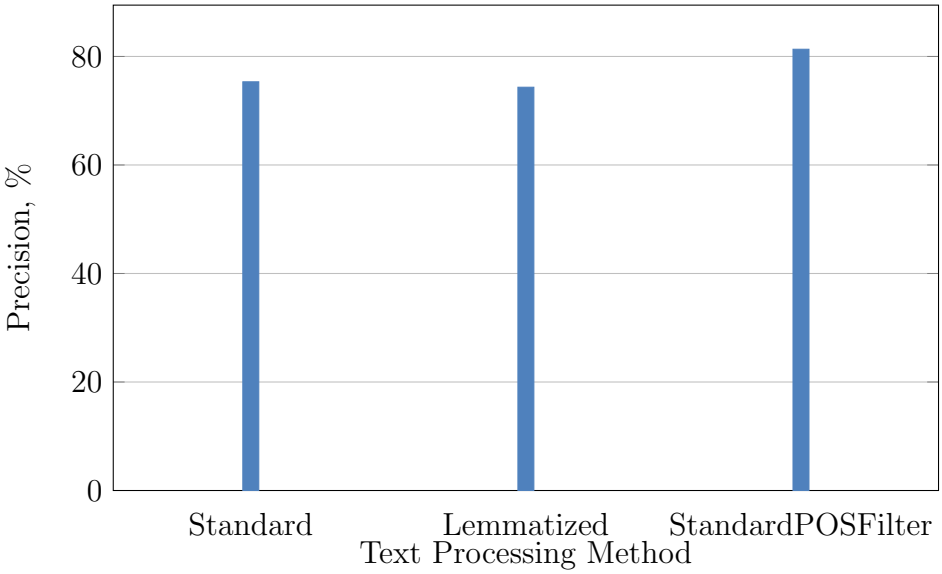

Figure 4.1: Precision for the results of experiment 1.

4.2 Experiment 2: Examining Number of Seeds

There are 6 different experiments are conducted in this part, where the number of seeds in the input list vary from 100 to 600 in steps of 100. The text processing method *"Standard"* is used for all experiments and the 100 most similar words are extracted. The number of drug names found in each experiment is shown in table 4.2, and the precision for each one of them is calculated and shown in figure 4.2.

	100 seeds	200 seeds	300 seeds	400 seeds	500 seeds	600 seeds
Standard	79	79	72	75	73	66

Table 4.2: Results for experiment 2 showing the number of drug names found for different number of input seeds.

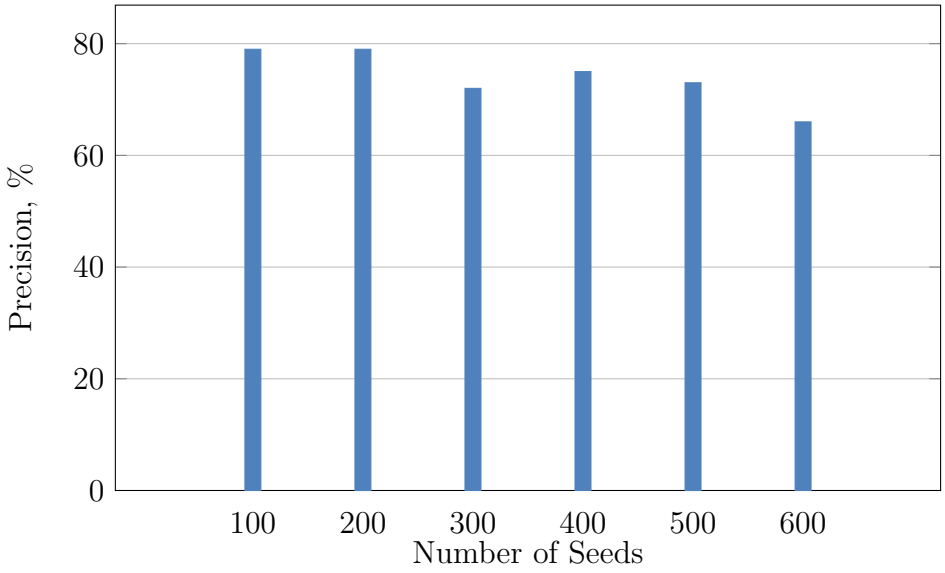

Figure 4.2: Precision for the results of experiment 2.

4.3 Experiment 3: Examining Extraction Criteria

In total 3 different experiments are conducted in this part where the number of similar words extracted vary from 100 to 300 in steps of 100. The text processing method *Standard* is used for all experiments and because the best result from previous experiment was achieved when using 100 or 200 seeds, 200 seeds were chosen as the number of seeds is inserted for these experiments. The number of drug names found in each experiment is shown in table 4.3, and the precision for each experiment is shown in figure 4.3.

	Extract 100 words	Extract 200 words	Extract 300 words
Standard	79	156	222

Table 4.3: Results for experiment 3 showing the number of drug names found extracting different number words.

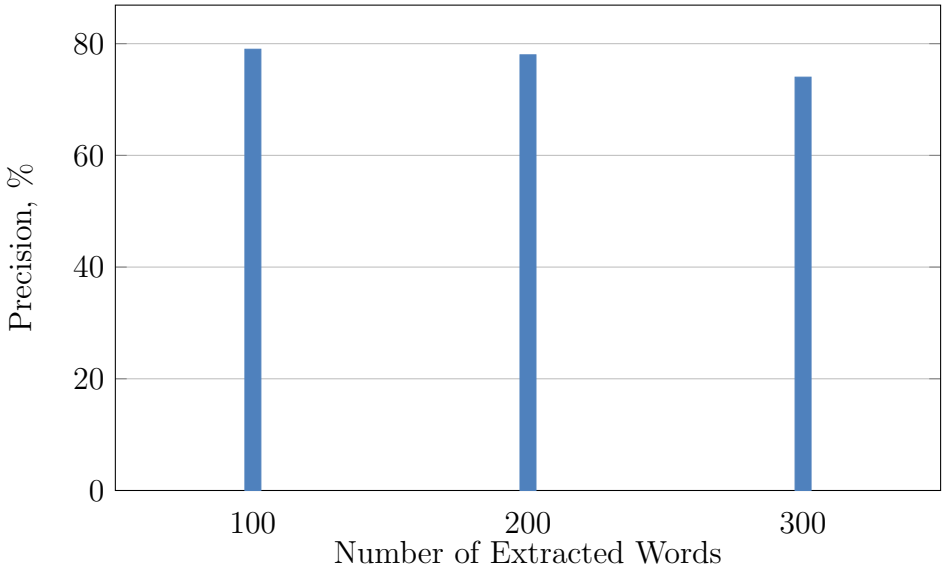

Figure 4.3: Precision for the results of experiment 3.

Chapter 5

Conclusion and Discussion

The results show that the method *"StandardPOSFilter"* had the best result out of all text processing methods, which indicates that most drug names are indeed POS-tagged as nouns. Using this method however poses the risk of filtering out words that are drug names but are labeled as something else than nouns. This would lead to there being more false negatives, something that the precision measure does not pick up. Because recall considers false negatives it could be an appropriate solution to this problem, and calculating it precision and recall could be balanced to find the optimal results. This may however be difficult to realize in practice if there is no prior knowledge of how many drug names are present in the input text.

The *"Lemmatization"* method got a slightly lower precision than *"Standard"*, which may be explained by the risk of loosing relevant information when lemmatizing text. This information could for example be what distinguishes the context of a drug name and a non-drug name, and after their contexts have been lemmatized, the algorithm might interpret them occurring in similar contexts and give them a high similarity scores.

In the second experiment the optimal result was achieved when using 100 or 200 seeds, so when few seeds were used the result got better. This may be explained by the possibility that using less seeds results in less non-drug names getting high similarity measures. According to the article [26], Ferret found that when using semantic similarity measures based on the Distributional Hypothesis high frequency words produce better results than low frequency words. In our case this would mean that high frequency seeds find more drug names than low frequency seeds. If for example a seed occurs only a few times in the whole text, its occurrences are unlikely to successfully capture its semantic meaning, which could lead to it getting high similarity measures with words of different semantic meaning (non-drug

names). The longer the seed list is, the more likely it is that it contains many low frequency words; and the more low frequency words present in the seed list, the more likely it is that non-drug names get high similarity measures. This may imply that the random indexing algorithm works best if only seeds known to occur often in the text are used.

In the third experiment it was shown that extracting the 100 most similar words resulted in a higher precision than extracting 200 or 300 words. It is reasonable to assume that the more words are extracted as result, the lower will be words' similarity score are and the less likely they are to be drug names. It is however also reasonable to assume that extracting more words can also lead to higher recall, and the results show that the more words were extracted, the more drug names were found. To find the optimal number of words to extract precision and recall should be balanced, but again it is difficult to calculate the recall since that requires for all drug names in the input text to be known.

Out of all experiments conducted the best result was found in experiment 1 when using the method *"StandardPOSFilter"*. When using the third seed list this experiment measured a precision of 86%, and the result list can be seen in Appendix A.

This project has shown that finding the names of drugs using the techniques random indexing and cosine similarity is possible and may be an appropriate method for future systems searching for new drug names on the Internet. The project focused on finding names of any drug, new and old, instead of just searching for new drug names. The reason for this is because there may not exist a lot of information about new drug names, making them difficult to evaluate. Instead it is investigated if the algorithm can find drug names with high precision, and if the precision gets high enough, the idea is that known drug names can be filtered out from the results leaving only unknown words that are highly likely to be drug names. Because these words are unknown they may be likely to be new drugs.

Chapter 6

Directions for Future Work

This project has focused on investigating the methods random indexing and cosine similarity to find semantic similarity between words in an input text. There are various different approaches that can be used for the same task and future work may include investigating the alternatives to see which ones produce the best results.

The preparation of the text before and after the extraction of semantically similar words may also be done in various ways. It is possible to both add and remove features to investigate which features provide the best results. An example of a feature that could be added is spell-checking words in the text in the preparation step. It is common that words are misspelled, causing the misspelled words to be interpreted as completely different words than the correctly spelled ones. This will cause these words to not describe the same context, even though they should, which in turn is likely to lower performance. For the after-processing the lists used to filter out drug related words could be changed and updated to make sure that no words easy to filter out are not present in the results.

In the implementation used for the experiments, only drug names of the length of one word can be found. However, it is often the case that drug names consist of multiple words and it would be of interest to develop a system also capable to retrieving multiple word drug names.

An interesting additional experiment would be to only use seeds that are known to occur often to see if they generate better results. Because using high-frequency words as seeds may produce better results [26], a seed list can be produced containing only drug names that have occurred at least a decided number of times.

When using the method developed in this project for finding new drugs problems arise due to the fact that the newer a drug is, the less times it has been mentioned. Because of this the statistics gathered for new drugs that have not been mentioned many times will probably not capture the semantic meaning very well making random indexing will perform worse. This will decrease the chances of the new drug getting detected, causing problems for early warning systems wanting to discover new drugs when they emerge. The more time passes, the more the new drug will have been mentioned, making it easier to detect but the process of finding it very slow. In this sense the task of finding new drug names is very different from the task of finding for example synonyms, and may be more difficult. Future work could include investigating how this issue can be solved and what features should be added to increase the chances of finding new drugs early.

Bibliography

[1] European Monitoring Center for Drugs and Drug Addiction. *New psychoactive substances in Europe, An update from the EU Early Warning System.* Luxembourg: Publications Office of the European Union (2015).

[2] Anticounterfeiting Committee - U.S. Subcommittee Public Awareness Task Force. 2015. *Report: Anticounterfeiting on the Dark Web.* http://www.inta.org/Advocacy/Documents/2015/ACC%20Dark%20Web%20Report.pdf (Accessed July 2015).

[3] He B, Patel M, Zhang Z, Chang K. C. *Accessing the Deep Web: A Survey.* Communications of the ACM 50, no. 5. (2007): 94-101.

[4] Spitters M, Verbruggen S, Van Staalduinen M. *Towards a Comprehensive Insight into the Thematic Organization of the Tor Hidden Services.* Proceeding '14 Proceedings of the 2014 IEEE Joint Intelligence and Security Informatics Conference. (2014): 220-3.

[5] Biryukov A, Pustogarov I, Weinmann R. P. *Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization.* Proceeding SP '13 Proceedings of the 2013 IEEE Symposium on Security and Privacy. (2013): 80-94.

[6] Van Hout M. C, Bingham T. *Silk Road, the virtual drug marketplace: A single case study of user experiences.* Int J Drug Policy 24, no. 5. (2013): 385-91.

[7] Griffiths P, Vingoe L, Hunt N, Mounteney J, Hartnoll R. *Drug Information Systems, Early Warning, and New Drug Trends: Can Drug Monitoring Systems Become More Sensitive to Emerging Trends in Drug Consumption?* Substance Use & Misuse 35, no. 68. (2000): 811-44.

[8] Sahlgren M. *An Introduction to Random Indexing.* Copenhagen: Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005, 2005.

[9] Gupta V, Lehal G. S. *A Survey of Text Mining Techniques and Applications.* Journal of emerging technologies in web intelligence 1, no. 1. (2009): 60-76.

[10] Marsland S. *Machine Learning: An Algorithmic Perspective.* CRC Press, 2009.

[11] Mili-Frayling N. Chapter 1. Text processing and information retrieval. In *Text Mining and its Applications to Intelligence, CRM and Knowledge Management.* Zanasi A. (ed.). Great Britain: WIT Press, 2005.

[12] Chakrabarti S. *Mining the Web: Discovering Knowledge from Hypertext Data.* Massachusetts: Morgan Kaufmann Publishers, 2002.

[13] Selkirk E, McCarthy J. J. Chapter 25. The Prosodic Structure of Function Words. In *Optimality Theory in Phonology: A Reader*, McCarthy J. J. New Jersey: John Wiley & Sons, 2004.

[14] Manning C. D, Raghavan P, Schtze H. Chapter 2. The term vocabulary and postings lists. *Introduction to Information Retrieval.* NY, USA: Cambridge University Press New York, 2008.

[15] Manning C. D, Raghavan P, Schtze H. Chapter 20. Web crawling and indexes. *Introduction to Information Retrieval.* NY, USA: Cambridge University Press New York, 2008.

[16] Sahlgren M. *Vector-based semantic analysis: Representing word meanings based on random labels.* Helsinki: Semantic Knowledge Acquisition and Categorisation Workshop at ESSLLI XIII, 2001.

[17] Sahlgren M. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* Ph.D. dissertation, Department of Linguistics, Stockholm University (2006).

[18] Dumais S. T. *Latent Semantic Analysis.* Annual Review of Information Science and Technology 38, no. 1. (2005) 189-95.

[19] K. Church, W. Gale. *Inverse document frequency (IDF): A measure of deviations from Poisson.* Text, Speech and Language Technology 11. (1999) 283-95.

[20] Manning C. D, Raghavan P, Schtze H. Chapter 6. Scoring, term weighting and the vector space model. *Introduction to Information Retrieval.* NY, USA: Cambridge University Press New York, 2008.

[21] Rong X. *Word2Vec Parameter Learning Explained.* (2014). arXiv:1411.2738 [cs.CL].

[22] Goldberg Y, Levy O. *word2vec Explained: Deriving Mikolov et al.s Negative-Sampling Word-Embedding Method.* (2014). arXiv:1402.3722 [cs.CL].

[23] Eugene F. K. *Taxicab Geometry.* Great Britain: Dover Publications, 1987.

[24] Sokolova M, Japkowicz N, Szpakowicz S. *Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation.* AI 2006: Advances in Artificial Intelligence 4304. (2006) 1015-21.

[25] Zhu X, Davidson I. *Knowledge discovery and data mining: challenges and realities.* PA, USA: IGI Publishing Hershey, 2007.

[26] Ferret O. Testing semantic similarity measures for extracting synonyms from a corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, D. Tapias. European Language Resources Association (ELRA), 2010.

[27] Gavagai. *Gavagai living lexicon.* http://lexicon.gavagai.se/ (Accessed 12 September 2015).

Appendix A

Result List for Experiment 1 StandardPOSFilter

premazepam	TRUE
pinazepam	TRUE
cinolazepam	TRUE
quazepam	TRUE
halazepam	TRUE
bretazenil	TRUE
methytrienolone	TRUE
amethyltestosterone	TRUE
ketazolam	TRUE
clotiazepam	TRUE
athleticism	
flutoprazepam	TRUE
loflazepate	TRUE
mda	TRUE
hydrocodone	TRUE
xtc	TRUE
nordazepam	TRUE
cipla	TRUE
sublingualtabletten	
overwhekming	
equivilancy	
cityblues	
oxycodone	TRUE
tapantadol	TRUE
nuxeum	TRUE
nitrazepam	TRUE
cola	TRUE

oxandrolone	TRUE
dapoxetine	TRUE
clonazepamflubromazepam	TRUE
mushroomtoads	TRUE
taboxycontin	TRUE
etizest	TRUE
loprazolam	TRUE
oxymetholone	TRUE
oxycontin	TRUE
usamg	
concertapills	TRUE
dexedrinedextroamphetamine	TRUE
doxymorphone	TRUE
tzf	
mgc	TRUE
topiramat	TRUE
mscontin	TRUE
ambien	TRUE
acetaminophen	TRUE
gear	TRUE
lincomycin	TRUE
indocin	TRUE
mdmamg	TRUE
sinemet	TRUE
motsvarar	
pharmavalium	TRUE
ecstacy	TRUE
oxycodones	TRUE
ket	TRUE
lamivudine	TRUE
ciplacarisoprodol	TRUE
eaacdaede	
ansaid	TRUE
thuoc	
cordarone	TRUE
mebeverine	TRUE
notasgood	
fosamax	TRUE
zanaflex	TRUE
stockvaltrex	TRUE
alfusin	TRUE

ezetimibe	TRUE
zeritstavudine	TRUE
imuran	TRUE
weed	TRUE
tramadol	TRUE
mesterolone	TRUE
hydrocodoneacetominophen	TRUE
cardura	TRUE
benazepril	TRUE
salbutamol	TRUE
warfarin	TRUE
alfacip	TRUE
enalapril	TRUE
colospa	TRUE
usdgrgr	
phenypropionate	TRUE
phyenylpropionate	TRUE
clonazeoam	TRUE
wyethpfizernitrazepam	
bromazepam	TRUE
eldepryl	TRUE
amlodipine	TRUE
combinaion	
omeprazoleprilosec	TRUE
lorazepam	TRUE
etodolac	TRUE
keflex	TRUE
intuniv	TRUE
acetazolamide	TRUE
plendil	TRUE
alprazolam	TRUE
hypnoticssleep	TRUE