Universitat de Barcelona

Facultat de Filologia

Unsupervised Vector Space Models for representing word meaning

(Master thesis)

Author: Venelin Kovatchev
Supervisors: M. Antònia Martí Antonín
Maria Salamó Llorente

Barcelona, 3rd of September, 2015

Contents

List of Figures

List of Tables

Abstract

The core hypotheses in Distributional Semantics is that the meaning of a word is closely related to its distribution and more specifically to the contexts it appears in. This approach to meaning can be computationally implemented by means of various statistical methods applied to a corpus of language use. Different mathematical frameworks can be used for this task. Some of the most productive ones rely on representing the word meaning as a vector in multi-dimensional space. The different models within these frameworks calculate the relationship between the vectors by means of different mathematical measures, such as the cosine of the angle between them or the linear distance. The hypothesis behind these frameworks is that this relationship between the vector representations can be used to make inferences about the relationship between the common-sense meaning of the represented words. The vector representations can be used to infer different types of relationship - vectors that are close to each other tend to be generated by similar words; pairs of vectors that have similar mathematical relationship tend to be generated by words that have similar common-sense relationship. In this thesis I focus on automated methods for obtaining semantically related words using vector space representations and unsupervised clustering of those representations.

I firstly replicate a methodology for extracting clusters of semantically related words from a corpus. This methodology has been successfully applied to Spanish (Marti et al. [2015]) and I do the replication with an English corpus. I discuss the necessary steps for using the methodology with a new language and analyze the obtained results. Secondly, I use a different methodology for obtaining vector representations and clusters of semantically related words, which has already been applied to English (Mikolov et al. [2013]). Finally I compare the representations and clusters obtained by the two methodologies. The two models differ in pre-processing requirements for the corpus, in computational complexity and execution time, in the information they extract, in the obtained results, and in their applicability. What is common for both is that they represent the meaning of a word as a vector in space, that they use the contexts a word appear in as dimensions of the space and that they include tools for clustering similar or related words.

Chapter 1

Introduction

1.1 From Cognitive science to Distributional Semantics

Contemporary cognitive science is complex and interdisciplinary - it aims to combine research in biology, linguistics, philosophy, and psychology. While these fields are different in many aspects, within cognitive science they have the same goal, to understand the way human cognition works. The knowledge of the human cognition can be used for many practical applications, such as: improving the process of learning, identifying and treating diseases, and creating automated tools for language processing. Most of the approaches to human cognition emphasize the importance of using of large amounts of empirical data and their objective processing. The development of the new technologies allows to gather data from many sources, for example fMRI can provide information about the brain activity, eye-tracking devices help analyzing attention, and measuring response time provides additional data in various cognitive experiments. These technologies attempt to objectively identify and classify patterns and regularities in human cognition based on actual human behavior.

Within linguistics, the focus is put on analyzing the nature, functioning and use of language. A common assumption within Cognitive Linguistics is that the knowledge of language can be found in the language use (Croft and Cruse [2004]). Tomasello [2000] suggests a model of acquisition based on language use and constructions. The empirical data used in this research direction are actual linguistic productions. As a result, many of the contemporary linguistic methods concentrate on working with corpora. In the modern era of the Internet, it is very easy to find a corpus of language use. The raw data can be extracted directly from the Web and the increased computational power of the computers facilitates faster and more powerful processing of the available data. Corpus linguistics use different

mathematical and statistical frameworks for obtaining valuable data from corpora. For example, Evert [2008] and Pecina [2010] present different statistical methods for extracting information from a corpus. They focus on identifying collocations by means of counting the co-occurrence in a corpus and applying different statistical algorithms to evaluate the degree of association between the words. The vector space models (VSM) I discuss in this thesis use more sophisticated computations and go one step further in the processing and analysis of the data. After determining the contexts of each word, the VSM use this information to generate a vector representation of the words. The important advantages of all of the methods based on corpora are the possibility to automatically process big amounts of data, and to interpret and analyze them objectively.

Using corpora in cognitive science research raises many questions, one of the most important being whether corpora contain valuable information about language and cognition at all. Arppe et al. [2010] answer this question and discuss the role of corpora in cognitive science from different perspectives. The authors analyze different situations where corpora can be used and compare approaches relying on corpora with other approaches, such as intuitions and data coming from psycholinguistic experiments. They raise some methodological questions related to corpus usage, but in general they are positive about the role of corpora as a source of information. They do argue, however, that the cognitive research should be multi-methodological and interdisciplinary and should not rely only on one source of information.

One of the central problems in cognitive science and in linguistics is the problem of mental representations, word meaning and reference. The problem, from a general cognitive point of view, is how do we make mental representations of the things of the world, how do we operate over these representations and how do we maintain a relation of reference with the world. In linguistics the problem is related to determining the meaning of the linguistic units - morphemes, words, and phrases - and the manner to represent them. Linguists need to define the meaning and also account for the relations between the meaning of different units (phenomena such as synonymy, antonymy, hypernymy, etc).

Many classical approaches to semantics rely on representing the meaning of a word using semantic primitives (Jackendoff [1990], Fodor et al. [1980]). The idea behind these approaches is that words can be represented using a closed set of (possibly non-linguistic) primitives. Winograd [1978] suggests a list of characteristics of the primitives and the systems based on them. With respect to primitives, he argues that the number of primitives should be smaller than the number of words, the different primitives should not be definable in terms of one another and each primitive should be irreducible. With respect to words, he points out that each word should be representable in terms of primitives, its representation

should be complete, that is, they should not leave any important properties of the word out. The representation also should be unique and unambiguous. There are different problems with the primitive based approaches and the most important one is probably how to determine the list of primitives. Winograd [1978] argues that in theory the list can be extracted empirically. However this does not happen in the actual applications of primitive based systems, where the list of primitives is usually predefined by the linguists. As a result, the representations are subjective and based on the intuition of the researcher. Fodor et al. [1980] raises philosophical arguments against the usage of primitives to define the word meaning. Boleda and Erk [2015] discuss some of the problems with primitive based approaches and compare them with the approaches based on Distributional semantics.

Distributional semantics take a different approach and follow the idea of representing each word in terms of its distribution in samples of actual language use. More specifically, the Distributional Hypothesis claims that the degree of similarity between two linguistic units is related to the similarity between the linguistic contexts they appear in. Lenci [2008] analyzes different aspects and approaches towards distributional semantics. From a formal point of view, he emphasizes the importance of defining "context". He distinguishes between "raw" and "syntactically savvy" contexts. "Raw" context simply means bag of words - words that appear next to each other in the corpus, regardless of order and syntactic relation, while "syntactically savvy" context includes syntactic properties. He also discusses the role of mathematics and statistics in determining the relevance of contexts and calculating the full distribution of each word. Unlike approaches based on primitives, distributional approaches to semantics do not require any predefined list of features, they extract the features from the empirical data: the features are the contexts. The generation of representations is objective and automated. Additionally, the formalisms used by Distributional Semantics represent the words in a quantitative manner and make it possible to compare the obtained representations.

A common way of formalizing the word representation in Distributional Semantics is using vectors in a multi-dimensional space. Linear algebra can be used to calculate the relations between the vectors. After that, clusters of related vectors can be created and these clusters should contain words which are semantically similar. Marti et al. [2015] follow these steps as part of a process of suggesting construction candidates. Their framework uses "syntactically savvy" contexts. Mikolov et al. [2013] suggests another method for obtaining vectors, using "raw" contexts. In this thesis I analyze both approaches and compare the representations they generate from corpora of different sizes.

1.2 Related work

The problem of defining the meaning of a linguistic unit (morpheme, word, phrase) and the way it refers to the world is not new. It has received a lot of attention in theoretical and empirical linguistics, in artificial intelligence, in psychology, and in philosophy. There are many hypotheses, but there is still none that can fully address all the questions and problems related to meaning, reference, compositionality, inference, and acquisition. From a computational point of view, the problem is further complicated by the need of choosing a proper way to represent the meaning and formalize it. A working model for representing meaning needs to address at least three questions – What is the unit that is going to be represented?; How can the meaning of that unit be formally represented?; How the formal meaning of one unit relates to the formal meaning of other units?. A complete model of computational semantics needs to account both for the nature of meaning and for the way it can be adequately represented. In this thesis I will concentrate on the mode of representation. The linguistic unit I will work with is the word.

Harris [1954] argues that *"each language can be described in terms of a distributional structure, i.e. in terms of the occurrence of parts relative to other parts"*. In his proposed distributional representation of the language, each element is represented in terms of its environments - all other elements it co-occurs with, with their respective positions. Following this hypothesis, the author proposes a methodology for describing languages based in distributional analysis. The language sample has to be separated into units and the distribution of each unit has to be calculated in terms of the contexts in which this unit appears. The distribution of all the units has to be compared and the similar ones have to be grouped together in classes of distribution. While the author agrees that distribution cannot be related to meaning in one-to-one manner, he argues that difference or similarity in the distribution of two elements often reflects certain facts about their meaning. More specifically, he argues that words with similar distribution have similar common-sense meaning.

At the time when Harris was formulating this hypothesis, he did not have the computational power nor the necessary corpora to fully explore and test it. However in recent years modern technologies allow for gathering and processing big amounts of linguistic data and the distributional approaches to semantics are becoming more popular. Lenci [2008] offers an overview of the aspects and tendencies of Distributional Semantics. From a methodological point of view he distinguishes between approaches that use "raw" contexts and approaches that use "syntactically savvy" contexts. The difference between the two is that the latter approaches use syntactic information when determining relevant contexts. The author formulates a generic Distributional Hypothesis that is applicable to all of the approaches - *"The degree of semantic similarity between two linguistic expressions A and B*

is a function of the similarity of the linguistic contexts in which A and B can appear". This hypothesis, inspired by the one formulated by Harris, suggests that a word can be represented in terms of its distribution. The hypothesis has the advantage that the representation can be formalized using mathematical and statistical methods and conclusions can be drawn from comparing the representations of different words.

As Lenci [2008] points out in his overview, the most common approach towards formalizing distribution is vector representation. In order to do that, first a **m** by **n** matrix has to be generated, to represent the relation between **m** words from the corpus and **n** contexts these words can appear in. Each of the **n**-dimensional vectors in this matrix represents the distribution of a linguistic unit (word) with respect to **n** different contexts. From this matrix, an **n**-dimensional "distributional space" is created and each of the units is represented as a point in this space. The proximity between different points in the distributional space can be calculated and used to draw conclusions about the semantic similarity of the words they represent.

Turney and Pantel [2010] present a survey of different Vector Space Models (VSMs) of semantics. They argue that the defining property of a VSM is the choice of the matrix that is used to define the dimensions of the distributional space. One source corpus can be used to generate multiple matrices depending on the choice of both the unit and the contexts. The authors describe three different matrices - term-document, word-context and pair-pattern. Term-document matrix represents the relation between a document and a list of terms. Such matrices can be used for document classification or web search. The two other matrices discussed in the article are centered around words. Word-context matrix represents the words with which each word co-occurs and measures what the authors call "attributional similarity". This is a direct application of Harris [1954] and measures word similarity based on similarity of contexts. Pair-pattern matrix represents the relation between a pair of words and the "pattern" in which these words appear in. The hypothesis behind such matrix is that pairs that appear within similar patterns tend to have similar relation. This matrix measures what the authors call "relational similarity".

Turney and Pantel [2010] point out that the choice of the matrix depends on the concrete task - different matrices provide different information. Baroni and Lenci [2010] present a more sophisticated approach - the information they extract from the corpus is not a matrix, but rather a third order tensor, with the dimensions representing "word", "link", "word". Adding another dimension allows for extraction and representation of more information. After that, from this single tensor, multiple matrices can be generated for specific tasks without the need of going back to the corpus. In the theoretical argumentation for their model, the au-

thors make an important distinction - "unstructured" versus "structured" models of Distributional Semantics. Unstructured models simply measure co-occurrence, without accounting for the type of relation between co-occurring words. Structured models take into account the syntactic relations between the co-occurring words. A structured model has more requirements for the source corpus - it needs to be tagged with morpho-syntactic information, however the authors argue that the information a structured module can extract is more reliable. The model they suggest is a structured one, it defines a word-link-word semantic space, with "link" being the syntactic dependency between the two words.

Erk [2012] presents a survey of different applications of VSM and the problems they have to deal with, most specifically the problem with polysemy. She analyzes a variety of different models for vector representations and discusses the possibility to generate vectors for whole phrases.

While the VSM can formalize word representations, they do not naturally include a methodology for grouping similar words together. Moisl [2015] discusses the applicabilities of cluster analysis for corpus linguistics. Cluster analysis performs partitioning of the available data based on relative similarity between the elements. Moisl [2015] argues that this is a good way to objectively (relative to the defined criteria) identify patterns in big sets of data. Furthermore he discusses multiple clustering algorithms designed to work with multidimensional vector representations.

Marti et al. [2015] present a method for *"automatic extraction of lexicosyntactic patterns that are candidates for consideration as constructions"*. The method uses structured VSM - the underlying matrix is a lemma-context matrix, where the token is replaced by it's lemma and the context is a combination of a syntactic dependency and a second lemma. This format is compatible with the work of Baroni and Lenci [2010], as this matrix can be obtained from a lemma-link-lemma tensor. After generating the matrix and calculating all the vectors, Marti et al. [2015] use CLUTO (Karypis [2003]) to combine lemmas that share a set of contexts to obtain clusters of semantically related lemmas. The obtained vector representations of the words and clusters of semantically related words are then used to generate construction candidates. Vectors and clusters are central part of the method - it is important that the vectors can be used to generate clusters and that the clusters contain words that are semantically related. The method reports good results, which suggests that the clusters obtained by using VSM represent relevant semantic information.

Mikolov et al. [2013] present a different approach for obtaining vector representations. They focus on the usage of Neural Network Language Models for generating vector representations and analyze the results and complexity of these models. In order to lower the computational complexity and to make it possible

to train the models on huge amounts of data (above 1 bilion words), the authors suggest two simplified models for obtaining vector representations - the Continuous Bag-of-Words Model (CBOW) and the Continuous Skip-Gram Model. The CBOW model uses neighboring words (both before and after the target word) to generate the vector, while the Skip-gram tries to classify each word based on each other word in the sentence. Both models do not use any kind of morpho-syntactic information about the words and are therefore similar to the unstructured models, discussed by Baroni and Lenci [2010]. The authors argue that their two models can obtain good vector representations and extract both syntactic and semantic information. They emphasize specifically the quality of the relational similarity that can be measured.

The VSM model used by Marti et al. [2015] and the models for obtaining vector representation suggested by Mikolov et al. [2013] are different in the preprocessing requirements for the corpus, in the algorithms that calculate the vectors and in the information they extract from the vectors. They also define a different semantic space. However both models report success in representing relevant semantic information and both models include algorithm for clustering semantically similar words together. One of the objectives of this thesis is to compare the way these models cluster together vectors obtained from the same corpora of language use.

1.3 Motivation

Throughout my bachelor degree I have been studying linguistics and cognition mostly from a theoretical point of view. However, I was also interested in working with empirical data and finding possible applications of the linguistic theories. The Cognitive Science and Language (CCiL) Master program gave me the opportunity to study the language from an empirical and interdisciplinary point of view. Arppe et al. [2010] point out that *"Ideally, research in cognitive linguistics should be based on authentic language use, its results should be replicable, and its claims falsifiable"* and I believe that this is the direction in which linguistics should work. This view is strongly empirical, as from one side it uses samples of language use as a source of information and from another it makes claims and predictions, which can be falsified using empirical data. Marti et al. [2015] present a model that goes in this direction. The authors extract information from a corpus of written language, make generalizations from the data, and propose a list of candidates to be constructions. As part of the process, they build up clusters of semantically related words, based on the proximity of their semantic vector representations. The results are verified for Spanish by native speakers. In this thesis I firstly replicate this process for English in order to determine whether it can also provide good results in another language and secondly I compare the representations and

their methodology for clustering with word2vec (Mikolov et al. [2013]), which is one of the well-known state-of-the-art approaches.

1.4 Objectives

This thesis has two main objectives. The first one is to replicate part of the work of Marti et al. [2015] until clustering process for English, to document the whole process and to evaluate the obtained results. The second objective is to compare the representations (vectors and clusters of vectors) with representations generated by word2vec from the same corpora.

1.5 Organization

The thesis is organized as follows: Chapter 2 explains the process of replicating Marti et al. [2015] in English. Chapter 3 explains the process of generating representations with word2vec. Chapter 4 introduces and compares the results from the experiments described in Chapters 2 and 3. Finally, the conclusions are presented in Chapter 5.

Chapter 2

Replicating Marti et al. (2015)

2.1 Introduction

The method proposed by Marti et al. [2015] has five main steps (see Figure 2.1). Step 1 is the linguistic processing of the corpus - the raw text is cleared from non-linguistic data, it is POS tagged and syntactically parsed. After that, during Step 2, a VSM matrix is constructed. The rows of the matrix correspond to lemmas and the columns correspond to contexts. Contexts in this approach are defined by combining a syntactic dependency and a lemma. This matrix is used to generate representations of the 10 000 most frequent words in the corpus. Next, Step 3 uses CLUTO toolkit (Karypis [2003]) to create clusters of semantically related lemmas from the VSM matrix and the corresponding vectors. The clusters are created based on descriptive contexts (contexts common for the words in the cluster) and discriminative contexts (contexts that differentiate the words in the cluster from the words outside of the cluster). Step 4 links the different clusters obtained in step 3. The linking is done using the most relevant descriptive and discriminative clusters. And finally, Step 5 filters the links between the clusters, removing relations that are not bidirectional and generates candidates to be considered as constructions.

From the implementation point of view, the different steps are separate scripts, each with its own input and output. The process is organized in such a way, that the output of one step is the input of the next one. With the exception of Step 1, the process is relatively language independent. As this thesis is centered around obtaining vector representations and the way these representations can be clustered, my work is focused on the first three steps. However, there are no methodological problems in continuing the replication with steps 4 and 5.

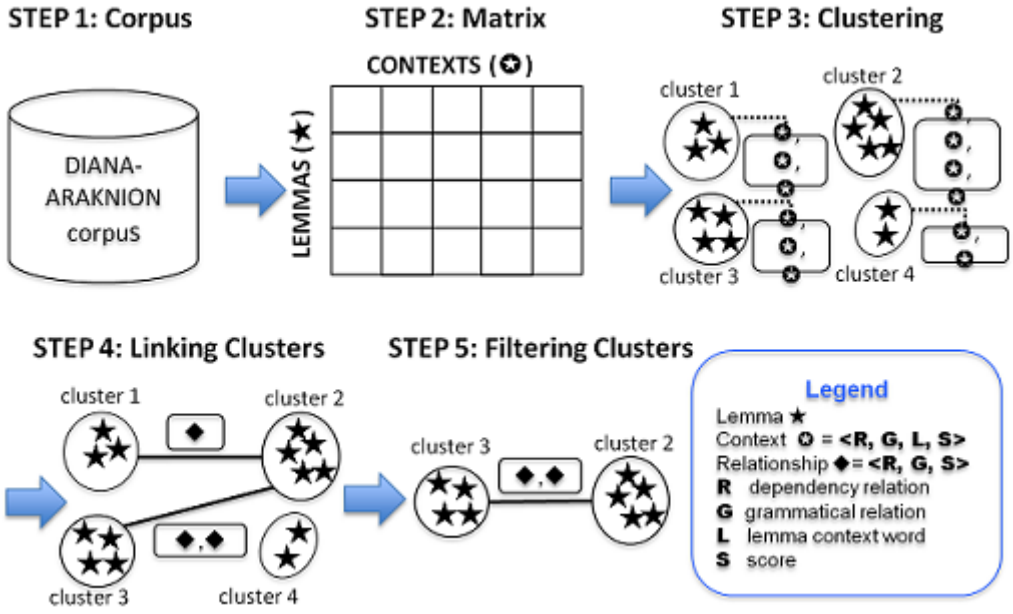

Figure 2.1: Figure 1 from Marti et al. [2015], pg 7

Step 1 of the process is language specific and cannot be replicated automatically. During this step, the corpus undergoes several modifications: all non-linguistic information (such as XML tags and special non-unicode symbols) is removed; the corpus is tokenized and all tokens and sentences are identified and indexed; the corpus is lemmatized, POS tagged and syntactically parsed. All these preprocessing steps are done taking into consideration the specifics of the language and the corpus used. When using a raw corpus, all of these modifications have to be made. When using a corpus that has already been processed in some way, some of the steps can be omitted and the format of the data adapted. At the end of Step 1, the corpus needs to have the following format:

Token	Lemma	POS	Short POS	Sent ID	Token ID	Dep ID	Dep Type
sanitarios	sanitario	NCMP000	n	0	2	27	suj

Table 2.1: Diana-Araknion Format

In this format, token is the word, the way it appeared in the corpus. Lemma is the automatically assigned lemma to the corpus. POS is the automatically assigned part of speech tag. Short POS is a short part of speech tag (see Section

2.3). Sent ID is the numeric unique ID of the sentence in the corpus. Token ID is the numeric unique ID of the token, corresponding to its position within the sentence. Dep ID is the numeric unique ID of the element with which the token has a syntactic relation. Dep type is the type of syntactic relation between the token and the Dep ID element. In the example provided the token is "'sanitarios"; the lemma that corresponds to it is "sanitario"; the part of speech tag is "NCMP000" (a type of Noun in the tagset used by Marti et al. [2015]); the short POS is "n" (noun); the Sent ID is 0, that is, the first sentence in the corpus; the Token ID is 2, that is, the second word of the sentence; the Dep Type and the Dep ID indicate that "sanitario" is a subject of token with id 27 (in this case, token 27 is "es", a form of Spanish verb "be").

As long as all the required information is available and presented in the corresponding format, the replication can proceed with Step 2 and generate a lemma-context matrix. However, the preprocessing of the corpus cannot be simply automated, as certain decisions need to be made. For English, the two most important preprocessing decisions are determining the size and content of the POS tagset and determining the type of syntactic dependencies that will be tagged. For this thesis I decided to use a corpus which is already preprocessed. However, it is necessary to modify the format, the POS tagset and the list of dependencies in order to meet the requirements of the next steps in the Marti et al. [2015] process.

The replication process is organized as follows. First I present and analyze the corpus that I am going to work with at Section 2.2. In Section 2.3 I describe the different formal modifications that I made on the corpus. In Section 2.4 I present the changes on the POS tagset and list of Dependencies. The corpus after all the described modifications is formally equivalent to Diana-Araknion corpus at the end of Step 1 of Marti et al. [2015]. Finally, in Section 2.5 I briefly comment on Step 2 and Step 3 and explain the different experiments that are to be run using different corpora size.

2.2 The corpus

The first step of the process of Marti et al. [2015] is the linguistic processing of the corpus. During this step, the corpus is tokenized, lemmatized, and POS and dependency tagged. To replicate the process, I had two options - either to start from raw corpus and apply all steps accordingly, or to use a corpus that has already been preprocessed. I decided to use an existing preprocessed corpus, the PukWaC corpus (Baroni et al. [2009]). It is a 2 billion word corpus constructed from sites in the .uk domain. The corpus, in the format it is available, is cleared from most of the XML tags, lemmatized, POS tagged with TreeTagger and dependency tagged with Malt Parser. I compared the corpus to the Diana-Araknion corpus

and determined that the information available (type and size of POS tags and Dependency tags) was similar enough for the purpose of replication. Additionally, the corpus preprocessing is within the standard conventions for English - the tagset used is based on the Penn Treebank tagset and the dependency structures are the ones from CoNLL-2008 SharedTask (Johansson [2007]).

The relevant technical specifications of the corpus are the following:

- there are few XML tags left, indicating the source URL of the different texts
- there are special tags to mark the start and the end of the sentence: "<s>", "<\s>"
- the information for each word includes the original token, the corresponding lemma, the POS tag, the syntactic dependency, the numeric unique ID of the token within the sentence, the numeric unique ID of the element with which the token has a syntactic relation and the type of the syntactic relation. A word in the corpus takes a format, as detailed in table 2.2.

Token	Lemma	POS	Token ID	Dep ID	Dep Type
Hooligans	hooligan	NNS	1	4	NMOD

Table 2.2: Wacky Format

In this example the token is "Hooligans"; the corresponding lemma is "hooligan"; the part of speech tag is NNS (singular common noun in the tagset used); the Token ID is 1; the Dep Type and the Dep ID indicate that "Hooligans" is a nominal modifier of token with ID 4 (in this case, token 4 is "passion").

- the POS tagset is extended version of the Penn Treebank tagset. It has 56 tags, as follows:
  - **$ " " ( ) , :** - tags for corresponding symbols
  - **CC** - conjunction
  - **CD** - cardinal number
  - **DT** - determiner
  - **EX** - existential "there"
  - **FW** - foreign word
  - **IN** - preposition

  - **JJ JJR JJS** - adjectives (general, comparative, superlative)
  - **LS** - list item marker
  - **MD** - modal auxilary
  - **NN NNS** - common noun (singular and plural)
  - **NP NPS** - proper noun (singular and plural)
  - **PDT** - pre determiner
  - **POS** - possessive particle
  - **PP PP$** - pronoun (personal and possessive)
  - **RB RBR RBS** - adverb (general, comparative, superlative)
  - **RP** - adverbial particle
  - **SENT** - sentence-final punctuation
  - **SYM** - symbol or formula
  - **TO** - infinitive marker
  - **UH** - interjection
  - **VB VBD VBG VBN VBP VBZ** - verb "be" (different forms)
  - **VH VHD VHG VHN VHP VHZ** - verb "have" (different forms)
  - **VV VVD VVG VVN VVP VVZ** - all other verbs (different forms)
  - **WDT** - wh- determiner
  - **WP WP$** - wh- pronoun (personal and possessive)
  - **WRB** - wh- adverb

- the syntactic Dependency list has 20 types of syntactic dependencies, as follows:

  - **ADV** - unclassified adverbial
  - **AMOD** - modifier of adjective or adverb
  - **CC** - conjunction
  - **CLF** - be/have in complex tense
  - **COORD** - coordination
  - **DEP** - unclassified relation
  - **EXP** - experiencer in specific cases
  - **IOBJ** - object

- **LGS** - logical subject
- **NMOD** - modifier of nominal
- **OBJ** - object
- **P** - punctuation
- **PMOD** - dependency between a preposition and it's child in PP
- **PRD** - predicative complement
- **PRN** - parenthetical
- **PRT** - particle
- **ROOT** - root clause
- **SBJ** - subject
- **VC** - verb chain
- **VMOD** - modifier of verb

2.3 Basic formal reformatting

As already mentioned in the previous section, the information available in the PukWaC is quite similar to the information contained in Diana-Araknion corpus. However it is not completely equivalent - Diana-Araknion is cleared from all non-linguistic information, while in PukWaC there are some XML tags remaining. Furthermore, there are additional variables that need to be calculated for each token in PukWaC (sentence id, short POS tag). Finally, the file has to be reformatted, using proper order of columns and sentence separation. Therefore it was necessary to modify the PukWaC corpus to let it match the format of the Diana-Araknion corpus. The formal modification of the corpus included four changes:

- The removal of all non-linguistic information from the corpus - XML tags and tags marking start/end of sentences. All non-unicode characters were also removed.
- Marti et al. [2015] assign a unique index to each sentence, and that index is included in the information for each of the words in the sentence. I indexed all the sentences and and added to each token the corresponding sentence id column.
- The generation of a short POS tag for each word in the corpus. The short POS tag is the first letter of each full POS tag (or first two when there are different parts of speech with the same first letter). It keeps the information

about the POS type, while discarding information such as verb tense or the difference between comparative and superlative. Both short and full tags are required by the process.

- The reordering of the columns, so they match the format of the Diana-Araknion corpus

For the purpose of the replication, I wrote a program in python, called WaCKy-modify.py. The program defines a class with the following functions:

- **split_line** - a function that identifies the different columns of the input file and puts them in separate variables
- **remap** - a function that identifies the different columns needed for the output and reorders all variables accordingly
- **gen_stag** - a function that generates the short POS tag. The map that indicates the full to short tag correspondence is user defined and can be modified.
- **mod_dep** - a function that modifies the syntactic dependencies. The map that indicates the old to new dependency correspondence is user defined and can be modified.
- **reformat** - a function that governs the process: reads input line, removes non-unicode characters, enumerates the sentences based on the start/end tags, calls the other functions in respective order and returns modified line in required output format.

The program has a separate config file, where the user can define multiple variables. He can define input and output format map - the type and order of the columns in the input and output file. The input map determines the format input file that the program will expect. The default one corresponds to the format described in Table 2.2, it expects a file with 6 columns with the following order: Token, Lemma, POS, Token ID, Dep ID, Dep Type. The output map determines the format of the output that the program will generate. The default one corresponds to the format described in Table 2.1, it generates a file with 8 columns in the following order: Token, Lemma, POS, Short POS, Sent ID, Token ID, Dep ID, Dep Type. Using the config file, WaCKy-modify.py can be used with different corpora formats.

The config file also includes a POS map and a syntactic dependency map. The POS map indicates the short tag corresponding to each POS. The default one is listed in Table 2.3. The syntactic dependency map indicates modifications that

have to be made in the list of syntactic dependencies. The default one is listed in table 2.4. The program can be invoked from a command line and takes the path to the input corpus as a parameter. It creates an object using the config settings, reads the corpus line by line and outputs the modified lines to a new file. The input and the output files have to be plain text files, with the different columns either tab delimited or space delimited.

After running WaCKy-modify.py, the PukWaC corpus was formally equivalent to Diana-Araknion corpus at the end of first step of Marti et al. [2015]. It has the same type of information for every word and the same number and order of columns. I verified the compatibility by testing the next steps of the process (generation of matrix and clustering) using a small sample from the PukWaC.

2.4 POS and Dependency tag modifications

Apart from the format of the corpus, I also revised all the existing POS tags and Dependency tags to determine if they were to be changed in any way. Marti et al. [2015] concentrate their work on arguments of the predicate and modifiers (nominal, adverbial, and adjective). Because of that, certain POS tags and Dependencies are not required and it is possible to simplify and optimize the process by removing or combining them. The changes I made in the corpus were in three directions. First, certain POS tags and syntactic dependencies were merged together (see Sections 2.4.1 and 2.4.3). Second, certain POS tags and syntactic dependencies were discarded as irrelevant in the replication process (see Sections 2.4.2 and 2.4.3). Third, the PMOD syntactic dependency between a preposition and its child in the prepositional phrases was reworked as a dependency between the child and the word the whole PP is attached to (see Section 2.4.4).

2.4.1 Merging the POS tags

The process of generating vectors, semantic space, and clusters is based on analyzing the contexts where each word appears in. A word is identified by its lemma and its POS tag. However, in the PukWac tagset there are POS tags which specify not only the POS tag of the token, but also contain information about other grammatical features, such as person, number, and tense. If these tags are kept unchanged, a separate vector will be generated for different forms of the same word, based on the different POS tag. For semantic generalization, generating different vectors for different word forms will affect the results negatively. In order to avoid that and to generalize the vector representation from all instances of the word, I decided to merge certain POS tags under one category. The following changes have been made (for a full list of POS changes see Table 1):

- **Adjectives** – the three types of adjective tags (general, comparative, superlative) were merged in one generic tag
- **Adverbs** – the four types of adverbial tags (general, comparative, superlative, particle) were merged in one generic tag
- **Nouns (common)** – the two types of common noun tags (singular and plural) were merged in one generic tag
- **Nouns (proper)** – the two types of proper noun tags (singular and plural) were merged in one generic tag
- **Verbs** – the 18 types of verb tags have all been merged in one generic tag. This includes the separate "be" and "have" tags.

The implementation of these changes was in function **gen_stag**. Marti et al. [2015] use only the short tag when calculating the vectors and matrices, therefore by mapping multiple POS tags to the same short tag means they will be treated the same way by the next steps of the process.

2.4.2 Discarding POS tags

In addition to the combination of specific tags in a more generic one, it is also necessary to discard some tags. Marti et al. [2015] focus on content word and as such, function words and punctuation do not carry much significant semantic information about the words they co-occur with. The following tags were marked as "other" in the corpus – *conjunction, determiner (general and pre), existential "there", foreign word, list item marker, possessive particle, pronoun (personal and possessive), number, symbol, infinitive "to", interjection, wh- words, punctuation.* The implementation was also in function **gen_stag**. All of the mentioned categories were mapped to short tag "o".

As a result of the changes introduced in 2.4.1 and 2.4.2 the tagset used in the experiment is modified as detailed in Table 2.3.

New tag	Original tag	Description
J	JJ JJR JJS	Adjective (all)
M	MD	Modal verb
N	NN NNS	Noun (common, all)
NP	NP NPS	Noun (personal, all)
R	RB RBR RBS RP	Adverb (all)
S	IN	Preposition
V	VB* VH* VV*	Verb (all)
O	CC CD DT PDT EX FW LS POS PP* SYM TO UH W* punctuation	Rest

Table 2.3: POS tagset modifications

2.4.3 Reorganizing dependencies

Similar to the POS tagset, the list of syntactic dependencies used in PukWaC is not fully relevant to the processing of Marti et al. [2015]. The focus on arguments of the predicate and modifiers means that certain dependencies should be discarded. While the unnecessary POS tags can lead to multiple vectors for the same word, unnecessary dependencies lead to a more complicated computational process for matrix and vector generation. More dependencies lead to more contexts, which are not too relevant to the information extracted. This increases the dimensionality of the vectors and of the semantic space and increases the computational cost. Therefore the modification of the dependencies is mostly related to the optimization of the computational process. After evaluating the existing dependency list, the following changes have been applied:

- the syntactic function **IOBJ** (indirect object) was merged together with the more generic **OBJ**. The PukWaC corpus has two types of "object" syntactic function - the "IOBJ" stands for indirect object, the "OBJ" is a generic object. While the distinction between direct and indirect object is important, it can be relevant only if direct and indirect object are clearly separated. The OBJ is in most of the cases Direct Object, however it also includes examples of Indirect Object. For this reason I decided to combine the IOBJ and the OBJ together.

- the following syntactic functions were discarded as not relevant to the current experiment: **CC** (conjunction), **CLF** (be/have in a complex tense), **COORD** (coordination), **DEP** (unclassified relation), **EXP** (experiencer in few very specific cases), **P** (punctuation), **PRN** (parenthetical), **PRT** (particle), **ROOT** (root clause).

The implementation of these changes is in function **mod_dep** - it rewrites the dependencies according to the user-defined map in the config. Dependencies that need to be discarded are rewritten as "_" in order to keep the same number of columns. As a result, the dependency list is modified as shown in Table 2.4.

Dependency	Description
ADV	Unclassified adverbial
AMOD	Modifier of adjective or adverb
LGS	Logical subject
NMOD	Modifier of nominal
OBJ	Direct or indirect object
PMOD	*see Section 3.4.4
PRD	Predicative complement
SBJ	Subject
VC	Verb chain
VMOD	Modifier of verb
empty	No dependency or discarded dependency

Table 2.4: Syntactic Dependencies modifications

2.4.4 PMOD dependency modification

The PMOD dependency in the corpus is a dependency between a preposition and its child in a PP. For example, in the phrase "I travel with guests", PMOD is the dependence between "with" and "guests". This dependency in most of the cases does not carry valuable semantic information. In this example, what could carry semantic information is the dependency between "travel" and "guests". In the syntactic dependencies used in the corpus,"with" is marked as VMOD of "travel". In order to obtain a dependency between "travel" and "guests" from the dependencies "with-guests" and "travel-with", I change the Dep ID of "guests" to the Dep ID of "with". I keep the original Dep type (PMOD), but after the modification, that dependency type is a dependency between the child in the PP and the word to which the whole PP is attached to. Marti et al. [2015] use similar approach towards the PMOD dependency in their work.

The implementation is done in the **mod_dep** function. The dependency map, apart from input-output correspondence, has a variable that indicates whether the dependency ID has to be modified. The default value of 0 does not modify anything. A value above 0 rewrites the dependency id to the dependency id of the parent. In the discussed example, a value of 1 changes the dependency "with-guests" to a dependency "travel-guests'.

2.5 Matrix and Vector generation

For the other two steps - Step 2 and Step 3 - I followed the approach of Marti et al. [2015]. The process of generating VSM is semi-automated. The script has to be executed manually and several parameters have to be set. I selected a minimum threshold of 5, same as the one used in the original experiment. All words that appear less than 5 times were ignored. From the words above this threshold, I generated vectors only for the 10 000 most frequent lemmas. This restriction was applied due to the computational time of the process. All of the words appeared in the contexts, even those with lower frequency. Similar to the original experiment, all proper names were replaced with the label "np". After the matrix was generated, I used CLUTO (Karypis [2003]) to automatically cluster similar vectors together. I generated different number of clusters (ranging from 100 to 1000) and used CLUTO's H2 metric to determine the optimal number of clusters, which has been 800 for all of the experiments. I then evaluated the semantic and syntactic similarities of the words which were grouped together manually.

If the process of Marti et al. [2015] is to be replicated completely, the clusters have to be used in Step 4 and Step 5. As the focus of this thesis is on vector representations and clustering, I did not replicate step 4 and 5 on all of the experiments. However, the obtained clusters are compatible with the requirements of Steps 4 and 5 and the full replication can use them.

When planning the part of the thesis related to Marti et al. [2015], I had at least two goals in mind: first, to determine whether the method they use is applicable in English (based on judging the semantic and syntactic similarity of the clustered words); and second, to see how the corpus size affects the quality of the vectors and the clusters. For this reason I generated and compared vectors and clusters from 5 different corpus samples - 1 million tokens, 2 million tokens, 4 million tokens, 20 million tokens and 40 million tokens. All corpora used is part of the PukWaC corpus (corresponding number of tokens from the start of the corpus). The results of the different experiments are described in Chapter 5.

Chapter 3

Generating vectors with Word2Vec

3.1 Introduction

Mikolov et al. [2013] present two methodologies for generating vector representations, based on deep-learning: Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model. Both classify the words and generate the vectors based on neighboring words, without taking into account any kind of syntactic dependencies or word order. CBOW model uses all the words that appear in a fixed window before and after the target word. Skip-Gram model tries to classify the word based on each single other word in the same sentence (within a fixed window of words before and after). The authors emphasize three important parameters that affect the quality of the vectors obtained - the size of the corpus, the size of the window, and the dimensions of the vector. The dimensions of the vector is the number of contexts that are used when generating the vector. For computational reasons, only fixed number of contexts are taken when calculating the representations.

In order to demonstrate their methodologies, the authors have made available a tool, written in C, named WORD2VEC[1]. The tool has to be downloaded and compiled on a 64 bit Linux distribution. It can then be run from the command line to generate and manipulate vector representations. All parameters, such as size of the vectors, the size of the context window and the learning algorithm can be specified when running the program. The tool takes raw text as input and generates vectors using either CBOW or Skip-gram algorithms. It can also group similar vectors together and generate a list of semantically related clusters.

The usage of WORD2VEC in my thesis is organized as follows. First I present

[1]It is freely available at https://code.google.com/p/word2vec/

the requirements for the corpus and the linguistic modifications I have made on it in Section 3.2. Then I describe the different parameters of WORD2VEC and the values I have used in the various experiments in Section 3.3

3.2 Corpus specifications

The input format that WORD2VEC works with is different from the formats of Diana-Arakion and PukWaC. WORD2VEC expects raw text - text without any kind of annotation. Therefore in order to use the tool I had to modify the available corpus and remove the existing annotation. For that task I created a python script, called WACKY-W2V.PY, which takes a text file in PukWaC format, discards all the information except the token itself and then appends the tokens one after the other. The script then outputs the new corpus to a text file. The output corpus contains no linguistic information and keeps the original order of the words in the corpus. It is compatible with the requirements of WORD2VEC. The default usage of WACKY-W2V.PY requires only path to the source corpus as a parameter, the script can be called from the command line.

For the basic usage of WORD2VEC, the default behavior of WACKY-W2V.PY was sufficient However, as certain linguistic information was available in PukWaC, I decided to expand the experiment and see if linguistic preprocessing can improve the results of WORD2VEC. The two algorithms (CBOW and Skip-Gram) generate vectors based on the words the way they appear in the corpus, therefore by changing the format of the tokens, I could change the representations.

I modified WACKY-W2V.PY and added three different output options: "basic", "lemma", and "pos". The basic output retains the original behavior, it keeps the token and discards all of the other information. The option for lemma output keeps the lemma instead of the token. It generates a corpus where each token is replaced by its corresponding lemma. The option for POS output keeps both the lemma and the short POS tag and generates a corpus where each token is replaced by its lemma and POS in a lemma_POS format. The short POS tag used here is equivalent to the short POS tag used in chapter 2.

WACKY-W2V.PY is run from the command line with two parameters: path to input corpus and output type (raw, lemma or POS). It outputs a corpus in the specified format. WORD2VEC can then be invoked to generate vectors from the newly created corpus.

There is a linguistic reasoning behind the decision to test the performance of WORD2VEC with lemmatized and POS tagged corpus. The program generates vector for every token in the corpus, based on the parameters and the methodology chosen. This means that from a raw corpus, it will generate different vectors for each different form. Singular and plural will have different vectors, present and

past of the verb will have different vectors, so will comparative and superlative of adjectives and adverbs. Some of the examples discussed by Mikolov et al. [2013] use these results. The authors demonstrate that this approach allows for calculating the relational similarity between pairs like "big-biggest" and "small-smallest". However I was interested in seeing whether combining all the forms of the words together can improve the vectors and their relations to different words as opposed to different forms of the same word. This is what the lemma output format can do, by replacing all word forms with the corresponding lemma I made sure I generate only one vector for the lemma. Adding the short POS tag in the pos output addresses the possible problem with homonyms, for example it differentiates between "bark_n" and "bark_v".

For the experiments with WORD2VEC I used the same corpora that I used when replicating Marti et al. [2015]: five corpus samples of different size. For each of the five samples, I used WACKY-W2V.PY to generate three different samples in WORD2VEC format - one raw corpus, one corpus in "lemma output" and one corpus in "pos output". I generated vectors for each of the samples, using the default CBOW algorithm. For the 40 million size corpus, I also used Skip-Gram for comparison. Mikolov et al. [2013] demonstrate results with different corpora size, and at 25 and 50 million tokens they already show relevant results.

3.3 Parameter selection

WORD2VEC has several command line options that need to be set in order to generate vectors and clusters. Apart from the path to input and output, the following options can be set:

- **size** – The number of dimensions for the vector. Default value is 100.
- **window** – The maximum skip length between words. Default value is 5.
- **sample** – The down sampling of high frequent words. Default value is 1e-3.
- **hs** – To use or not hierarchical softmax. Default value is 0 (disabled).
- **negative** – The number of negative examples. Default is 3.
- **threads** – The number of threads to be run simultaneously. Default value is 12.
- **iter** – The number of training iterations. Default value is 5.
- **min-count** – The minimal occurrence threshold to generate vector for a word. Default value is 5.

- **alpha** – The initial learning rate. Default value is 0.025 for skip-gram and 0.05 for CBOW.
- **classes** – The number of clusters to be generated. Default value is 0 (disabled).
- **cbow** – The learning algorithm to use. Default value is 1 (CBOW). 0 has to be set for Skip-Gram

As part of the program, there is example script for cluster generation with recommended values, which slightly differ from the default ones - the window size is 8, the number of negative examples - 25, the number of training iterations - 15. Mikolov et al. [2013] discuss the importance of the vector size and demonstrate that increasing the size improves the results. For the corpora of size 25-50 million they demonstrate significant improvement when increasing the size from 100 to 300, however increasing the size further is ineffective. Taking these results in mind, I decided to use size of 400 dimensions. The biggest problem with selecting parameters was choosing the number of clusters and the min-count. WORD2VEC does not have option to only work with a fixed number of words - it generates vectors for all words in the corpus that are above the selected threshold. It also does not have a mechanism to determine optimal number of clusters. I had two options - either to generate vectors for all of the words and choose a higher number of clusters or to artificially lower the number of words by increasing the threshold, so only 10 000 would meet the requirement. I ran both experiments, as described with more details in Chapter 4.3. The parameters I used in the end were as listed in table 3.1:

Parameter	Value
size	400
window	8
sample	1e-5
hs	0 (disabled)
negative	25
threads	1
iter	15
min-count	*varies*
classes	*varies*
cbow	1 (yes)
alpha	*default*

Table 3.1: Word2Vec parameters

The final command that I used to generate vectors had the following syntax:

```
word2vec -train corpora/40mil-raw.txt \\
-output clusters/sg-w40m-class.txt -cbow 1 -size 400 \\
-window 8 -negative 25 -hs 0 -sample 1e-5 -threads 1 \\
-iter 15 -classes 3000
```

The results of all of the experiments, as well as the values for min-count and classes are described in Chapter 4.

Chapter 4

Analysis of the results

In this chapter I present the results of the different experiments described in Chapter 2 and Chapter 3. This chapter is organized as follows - Section 4.1 describes the setup of the experiments and the means to compare the results. Section 4.2 presents the results of the replication of Marti et al. [2015]. Section 4.3 presents the results of using Word2Vec. Section 4.4 presents comparison between the results.

4.1 Setup

This thesis has two main objectives, as defined at Section 1.4 - to replicate the work of Marti et al. [2015] for English and to compare the obtained representations with representations generated using Word2Vec. Additionally, several other questions were raised when preparing the experiments. I was interested in determining the minimal corpus size required to obtain good representations with the different methodologies. I also wanted to see if using lemmatized and pos tagged corpus could improve the results of Word2Vec. In order to achieve these goals, I ran 41 different experiments: 5 using the methodology of Marti et al. [2015] and 36 using Word2Vec.

The replication of Marti et al. [2015] consisted of 5 experiments, using 5 different corpora sizes - 1M, 2M, 4M, 20M, and 40M. The original experiment uses a corpus of Spanish with the size of approximately 100 million tokens. The methodology and the steps of the replication were described in Chapter 2. The process is well described and standardized and as such there were no differences in the 5 experiments, apart from the corpus size. In all of the cases lemmas that appeared less than 5 times in the corpus were discarded. From the remaining lemmas vector representations were generated only for the 10 000 most frequent ones. Each of the corpora used consists of different lemmas with different frequency, therefore the 10 000 most frequent lemmas are different for each corpus. The number of

lemmas (10 000) is the same for all of the experiments and is the same as the number used by Marti et al. [2015] for 100M. It is determined by computational reasons - raising the number of lemmas would increase the time CLUTO needs for calculating the vectors. The optimal number of clusters was calculated using CLUTO's H2 and set to 800, slightly different from the optimal number for for Diana-Araknion corpus, which which was calculated to be 700. All of the clusters were generated using CLUTO (Karypis [2003]).

The multiple experiments with Word2Vec used the same corpora samples as the replication of Marti et al. [2015]. Following the modifications described in Chapter 3.2, three different corpora were created from each one of the 5 source corpora - a raw corpus, a lemma corpus, and a lemma-pos corpus. Due to the problems with determining the optimal number of clusters described at 3.3, I ran two different sets of experiments with each of these corpora.

In the first set of experiments I generated vectors only for the most frequent 10 000 lemmas. In the case of raw text corpus, the number was 15 000, due to multiple word forms counting as separate words. The words were grouped in 800 clusters (1200 for the raw text). Since Word2Vec does not have an option to select only the most frequent words, I applied the restriction by increasing the occurrence threshold as shown in table 4.1. All of the thresholds were determined empirically, in order to generate vectors only for the 10 000 most frequent lemmas.

Corpus size	Threshold for raw corpus	Threshold for lemmatized corpus	Threshold for lemmatized and POS tagged corpus
1M	5	5	5
2M	8	10	11
4M	15	20	22
20M	75	90	100
40M	150	180	200

Table 4.1: Occurrence threshold for different corpora

In the second set of experiments I used the default behavior of Word2Vec and generated vectors for all words that appear in the corpus more than 5 times. I determined the number of clusters manually, relative to the number of lemmas. For corpora sizes of 1M, 2M and 4M I used 800 clusters, the same number as Marti et al. [2015]. I used 2000 clusters for the 20M experiments and 3000 for the 40M.

In each of the two sets of experiments, 15 experiments were ran using the CBOW algorithm - 3 experiments (raw, lemma, lemma and pos) for each corpora size (1M, 2M, 4M, 20M, 40M). 3 control experiments were run with the 40M cor-

pora using the Skip-Gram algorithm. This made 18 experiments per set and a total of 36 experiments using Word2Vec. The parameters used for vector generation are described at 3.3

The goals of the thesis require comparison between clusters obtained in different experiments, therefore I had to suggest a methodology for that comparison. At the moment, there is no external resource which can be used to objectively evaluate the semantic relatedness of a cluster. The mathematical frameworks available for cluster validation can only be used to determine the validity of the cluster selection relative to the chosen formalism - whether the number and the content of the cluster represents valid structures within the data. For the purpose of this thesis I will assume that the clustering algorithms do generate valid clusters (from a mathematical point of view). This assumption also means claiming that the quality of the clusters is mostly depending on the quality of the vector representations. If the obtained clusters contain semantically related words, this means that the representation successfully catches aspects of the common sense meaning of the words. However, as there are no external resources or mathematical tools for evaluating the common-sense semantic relatedness of the cluster, the evaluation of the clusters was done manually. All of the clusters were evaluated based on two criteria: POS coherence of the cluster and semantic relatedness of the words. Four clusters will be presented and analyzed in this thesis from each experiment. In particular those, which contain the words “bicycle”, “child”, “give”, “calculate”. The full list of clusters is available at http://venelin.support.bg

4.2 Results with CLUTO

In this section I present the results obtained using CLUTO.

4.2.1 1M and 2M corpus

The quality of the obtained vector representations is very dependent on the size of the corpus used. Increasing the corpus size provides more relevant contexts and is expected to provide better representations. The first two experiments, using 1M and 2M token corpora, aimed at determining how the method works with relatively small corpus. The expectations for these experiments were that the quality of the clusters would be low. The obtained results are the following:

1M corpus

Cluster 1: *bicycle_n, bombard_v, bottom_n, bubble_n, buck_n, cable_n, capacity_n, cardsnow_n, cattle_n, caught_n, caution_n, ceo_n, circumstance_n, client_n*

Cluster 51: *child_n, cart_n, charm_n, chat_v, clamber_v*

Cluster 198: *give_v, compouns_n, contempt_n, elimination_n, gramophone_n, invoke_v, kid_n, km_n, loss_n, mac_n*

Cluster 97: *calculate_v, beseem_v, bios_n, blueprint_n, boiler_n, bollard_n, borrowing_n, boyfriend_n, "buffalo"_n, bundist_n, burial_n, burst_v, butcher_n, button_n, calgary_n, calibration_n, canoe_v, cash_n, catholicism_n, celt_n, chamber_n, chemo_n, chipset_n, chocolat_n, citation_n, classification_n, code_v, garment_n, gear_v*

2M corpus

Cluster 223: *bicycle_n, allegation_n, fancy_n, fervour_n, grandson_n, prerequisite_n, repossession_n, sliproad_n, temptation_n, tin_n, turn_n*

Cluster 282: *child_n, individual_n, man_n, people_n, person_n, student_n, user_n, woman_n*

Cluster 683: *give_v, bring_v, find_v, get_v, include_v, keep_v, make_v, offer_v, provide_v, see_v, show_v, take_v, use_v*

Cluster 665: *calculate_v, 90grams_n, choreographer_n, compute_v, resonate_v, run-up_n, scarab_n, shoot_v, turnaround_n, workspace_n*

The first conclusion that can be made is that even extracted from such small corpus, the clusters are relatively POS coherent. The clusters obtained from the 1M corpus are less POS coherent: Cluster 51 has 3 nouns and 2 verbs, Cluster 198 contain mostly nouns, however it also contain "give" and "invoke". The clusters obtained from the 2M corpus are significantly better at POS coherence, although there are still some mixed clusters, such as 665. The second conclusion is that the clusters obtained consist mostly of semantically unrelated or weakly related words. In the case of clusters obtained from 2M token corpus, some semantic relations can be found. Cluster 282 actually shows good semantic relation, as it contains only individuals. Cluster 683 contains semantically related verbs "give", "take", "get", "provide", and "offer" as well as some verbs, which are not that directly related. Overall, the semantic relatedness of the words in the clusters both in the experiment with 1M corpus and with 2M corpus is not clear enough.

The conclusion from these two experiments confirms the initial expectations - the clusters obtained cannot be considered good and strongly related. However, improvement both in POS coherence and in semantic relatedness is noticed with the increase of corpus size. Even at corpus size as low as 2M tokens, some of the clusters contain semantically related words.

4.2.2 4M corpus

The expectations for the experiment with 4M corpus were not much different from the expectations for the experiments with 1 and 2M. However, the obtained results were slightly better than expected:

Cluster 560: *bicycle_n, beacon_n, bucket_n, dispatch_n, diy_n, handbag_n, mast_n, mission_n, ou_n, stalk_n, torch_n, warhead_n*

Cluster 17: *child_n, adult_n, family_n, individual_n, man_n, member_n, people_n, person_n, pupil_n, student_n, woman_n, worker_n*

Cluster 70: *give_v, add_v, bring_v, find_v, get_v, keep_v, make_v, offer_v, provide_v, put_v, receive_v, see_v, send_v, take_v, use_v*

Cluster 461: *calculate_v, adjust_v, affect_v, control_v, depend_v, derive_v, differ_v, estimate_v, influence_v, measure_v, predict_v, vary_v*

Once again, the obtained clusters are POS coherent. At the same time, the degree of semantic relatedness seems to be higher than the one noticed in the experiment with 2 million token corpus. Cluster 560 is still not good, however at least some of the words seem to be related: "beacon", "dispatch", "mission", "stalk", "warhead" can all be used in military topics. Cluster 17 is good, it contains the words from Cluster 282 in the 2M experiment and it has added more related words to the list. Cluster 70 is better compared to Cluster 683 from the 2 mil experiment, as it removes the unrelated word "include", while adding "receive" and "send". However, the cluster is still not good enough, with words such as "use", "see", "put". Cluster 461 is significantly better compared to Cluster 665 from the 2M experiment - about half of the corpus is of semantically related words, for example "calculate", "derive", "estimate", "measure", "predict". Some of the other verbs in the cluster are also related, for example "affect", "control", "depend", "influence". While it is hard to generalize over the whole cluster, there are clear relations between some of the words.

The results from this experiment suggest that at around 4 million corpus size, the methodology starts generating clusters of words that are mostly POS coherent and have some degree of semantic similarity between the words. The results are still far from clear and the semantic relatedness is often between some of the words, but not all. However, it is clear that there is some relatedness for a big part of the clusters, which is not true for the clusters obtained from 1 and 2M token corpora.

4.2.3 20M corpus and 40M corpus

After the experiments with 4M corpus showed some relevant semantic information, I continued with two experiments using more realistic coprus size: 20 and 40 millions corpus samples. The expectations about those experiments were to obtain good clusters, not only compared to the clusters obtained in the 4M experiment, but in general.

20M corpus

Cluster 108: *bicycle_n, bike_n, cycle_n, motorbike_n, motorcycle_n, motor_n, scooter_n, tandem_n, tractor_n, trailer_n, wheelchair_n*

Cluster 3: *child_n, adult_n, individual_n, learner_n, other_n, parent_n, participant_n, patient_n, people_n, person_n, pupil_n, resident_n, student_n, woman_n*

Cluster 149: *give_v, add_v, bring_v, call_v, contain_v, enjoy_v, feature_v, find_v, follow_v, get_v, include_v, keep_v, make_v, offer_v, provide_v, see_v, set_v, take_v, use_v*

Cluster 390: *calculate_v, account_v, amount_v, average_v, calculate_v, double_v, equate_v, estimate_v, exceed_v, halve_v, lower_v, multiply_v, subtract_v, total_v*

40M corpus

Cluster 347: *bicycle_n, automobile_n, bike_n, driving_n, endurance_n, motorbike_n, motorcycle_n, motor_n, off-road_n, racing_n, scooter_n, tractor_n, wheelchair_n*

Cluster 220: *child_n, adolescent_n, adult_n, baby_n, child_n, infant_n, occupant_n, patient_n, sufferer_n, survivor_n, teenager_n, victim_n, woman_n*

Cluster 247: *give_v, attract_v, bring_v, deliver_v, draw_v, follow_v, introduce_v, meet_v, offer_v, present_v, provide_v, raise_v, represent_v, set_v, support_v, welcome_v*

Cluster 390: *calculate_v, control_v, decrease_v, double_v, estimate_v, increase_v, limit_v, lower_v, measure_v, minimise_v, minimize_v, offset_v, predict_v, reduce_v*

Overall, the results obtained from these two experiments are good. The clusters are strongly POS coherent and the words in them are mostly semantically related. The results are also evidently better than the ones obtained in the 4M corpus experiment. The clusters containing “bicycle” (108 from the 20M corpus and 347 from the 40M) consists mostly of vehicles. These clusters are bad in all previous experiments. The clusters containing “child” once again contains only

persons. However, cluster 220 from the 40M experiment shows stronger relation, containing mostly age-related words for persons, such as "baby", "adolescent", "infant", "teenager", and "adult". The clusters containing "give" both in the experiment with 20M and in the one with 40M do not provide too good results. While there are some related words ("give", "take", "get", "offer", "provide" in the 20M experiment and "introduce", "meet", "present", "welcome" in the 40M), the overall semantic relatedness of the clusters is not strong enough. These results suggest that the methodology might experience more problems when classifying verbs compared to when classifying nouns. The clusters with "calculate" however contain strongly related words and demonstrate that verb clusters can also be good.

The results from these two experiments clearly show that the methodology can be used for obtaining clusters of semantically related words in English. At least in the discussed cases, there seem to be no significant difference between the clusters obtained from 20M corpus and from 40M corpus. However, increasing the size of the source corpus to 100 million tokens and above could lead to a more noticeable improvement of the results.

4.2.4 Summary

Overall, the experiment with CLUTO was successful. The results of the five experiments show that the methodology is applicable for English. The experiments with corpus of 1 and 2 million tokens obtained clusters, which are POS coherent, however they did not succeed in obtaining relevant semantic classification. The experiment with corpus of 4 million tokens also provided POS coherent clusters, but also successfully managed to extract at least some semantic information. The results from the experiments with 20 and 40 million token corpora provided results, which are good with regard to both criteria. Increasing the corpus size from 1 to 20 million appears to significantly improve the results. Increasing the size from 20 to 40 mil does not seem to matter that much, at least in the example cases discussed. However, as the 20 million corpus already provides good results it is harder to note a vast improvement. When evaluating the quality of the clusters, it was noticed that noun clusters are more semantically related than verb clusters, at least until 40M token size of the corpus.

4.3 Results from Word2Vec

In this section I present the results obtained using Word2Vec. I first present the results from the set of experiments using all of the words. Then I present the results from the set of experiments where only the most frequent 10 000 lemmas

(15 000 words) were used.

4.3.1 1M and 2M corpuses (all words)

The expectations for the experiments using corpora of 1M and 2M were not high. The authors of Word2Vec demonstrate results starting from 25M and 50M corpora. The goal of the experiments was to determine whether any information can be extracted from smaller samples. The obtained results were the following:

1M corpus

Raw text

The token "bicycle" did not appear at least 3 times in the corpus

Cluster 389: *child, abuse, accumulated, age, birth, doctor, healthy, illness, murder, pain, parent, parents, pregnancy, pregnant, refuse, surgery*

Cluster 257: *give, advise, alert, apply, assist, confirm, consult, disclose, engage, give, hold, include, inform, inspect, introduce, involve, notify, possess, receive, refer, require, send*

Cluster 622: *calculate, arithmetic, calculate, clock, coherent, compress, container, creates, desired, disadvantage, dragging, exclude, filter, formula, frequencies, lacking, newer, notation, packet, pair, pattern, produces, prototype, rendering, responsive, simple, simpler, simplest, test, thin, trait, understated, varies, watchmaker*

Lemmatized text

Cluster 756: *bicycle, bike, biking, cabin, crew, Leasingham, quad, ride, riding*

Cluster 113: *child, age, birth, physically*

Cluster 86: *give, restrict, shop*

Cluster 194: *calculate, allow, correct, probability, sensible*

Lemmatized and POS tagged text

Cluster 384: *bicycle_n, accessory_n, Adams_np, Aluminum_np, Assorted_np, badge_n, biking_n, bra_n, Coin_np, Colour_np, Contains_np, Create_np, Crystal_np, devotion_n, Discover_np, Easy_np, elastic_j, Express_np, Fashion_np, Favourite_np, Fender_np, Flower_np, Funky_np, Fun_np, gem_n, Glue_np, greeting_n, Henna_np, Instructions_np, Kit_np, LA_np, lie_n, Mailorder_np, Make_np, Maker_np, Metal_np, Needle_np, nicely_r, Paintbrush_np, Paint_np, pastry_n,*

personalise_v, plaster_n, Plastic_np, polish_v, punch_v, RADEON_np, Skin_np, Sound_np, super_j, Thai_np, Titanium_np, tooth_n, Umbrella_np, uniform_j

Cluster 112: *child_n, adult_n, age_n, baby_n, birth_n, carers_n, childhood_n, mother_n, parent_n, relative_n*

Cluster 81: *give_v - the only member of the cluster*

Cluster 459: *calculate_v, balance_v, capture_v, detect_v, disadvantage_n, discrete_j, generate_v, handle_v, ie_n, interpret_v, key_v, load_v, manipulate_v, master_v, measure_v, numerical_j, occur_v, parameter_n, scale_v, sensitive_j, shape_v, spawn_v, store_v, vary_v, verify_v*

2M corpus

Raw text

Cluster 77: *bicycles, afternoons, bars, bingo, breaks, cafes, Calvi, catering, clubs, fairs, festivals, galleries, golf, gym, holiday, holidays, hotels, indoor, occasional, outdoor, outdoors, pitches, pool, premier, pub, pubs, pursuits, restaurant, restaurants, riders, sauna, shopping, shops, ski, stalls, surf, swimming, taster, tennis, touring, trips, Valkenburg, vans, venues*

Cluster 400: *child, activity, adult, assistants, bullying, childhood, children, classroom, esteem, helpers, intervention, junior, nursery, parent, parental, Parental, parents, peer, pupil, teacher, teachers, worker*

Cluster 244: *give, giving, offer, provide*

Cluster 710: *calculate, accurately, actual, alter, assumes, attain, avoided, balanced, calculate, calculation, certain, complexity, conventional, define, depend, depends, describe, desirable, determine, disadvantage, energies, explicit, extent, Furthermore, geographical, hence, implicit, implied, intelligent, interpret, interpreted, obsolete, occurrence, orientation, particular, preferences, prevents, probabilities, querent, react, relate, reverse, satisfactory, satisfies, Such, trajectory, whereby, which*

Lemmatized text

Cluster 619: *bicycle, acrylic, Alpine, antique, arc, Bach, Bedford, butterfly, Cafe, caricature, Cave, Clement, cookery, dancer, Denny, dinosaur, doorstep, eclectic, en, extensively, Gaelic, Gogh, mediaeval, Memories, miniature, painting, Picasso, pictorial, Pieter, satirical, sculpture, Thai, Told, Van, Vinci, watercolour, watercolours, Whiteread*

Cluster 108: *child, adult, lone, nursery, parent*

Cluster 228: *give, answer, ask, Frequently, interviewer*

Cluster 680: *calculate, aggregate, aircrete, axis, bulk, calculation, canvass, considerably, corresponding, cumulative, deviance, deviation, differential, diffraction, eigenvalue, eigenvectors, estimation, Figure, fluctuation, fraction, frequency, Ga, gauge, grouping, indicator, latitude, likelihood, magnitude, metric, mosaicity, multiply, overlap, partial, PDM, phi, plume, proportional, quantity, radius, randomly, sampling, scale, scaling, turbulence, turbulent, variant, volume, yield, zero*

Lemmatized and POS tagged text

Cluster 338: *bicycle_n, accessory_j, amusement_n, antenna_n, assortment_n, attic_n, backpack_n, bedding_n, bouquet_n, box_v, bracelet_n, bra_n, broom_n, caravan_n, carry_n, catcher_n, cellar_n, champagne_n, charm_n, cleaner_n, clip_v, clothes_n, combo_n, cover_n, deck_n, decorate_v, decoration_n, decorative_j, dispatch_n, don_v, elegantly_r, embroider_v, fashionable_j, finish_n, flat_r, glamorous_j, glitter_v, glitzy_j, Gym_np, hallway_n, hanger_n, hook_v, ignite_v, jacket_n, jack_n, jean_n, leather_n, magnet_n, mini_j, nail_n, nicely_r, obscure_v, Olive_np, paint_v, pickle_n, pierce_v, polish_n, pom_n, postman_n, purse_n, rack_n, refrigerator_n, retro_n, roller_n, satin_n, shelf_n, shoe_n, silk_n, silver_j, silver_n, soap_n, sock_n, sparkle_n, special_n, squeeze_v, super_j, Tesco_np, timer_n, trouser_n, tub_n, Umbrella_np, underwear_n, vase_n, wardrobe_n, wash_n, wooden_j, wrap_n, wrap_v, wrist_n*

Cluster 730: *child_n, adult_n, aim_v, attainment_n, bridge_v, bsm_n, buddy_n, bully_v, childcare_n, compulsory_j, deprive_v, disabled_j, disadvantaged_j, esteem_n, gifted_j, helper_n, junior_j, literacy_n, motivated_j, NLS_np, numeracy_n, Ofsted_np, old_n, parental_j, parent_n, phonics_n, primary_n, Primary_np, pupil_n, schooling_n, school_n, secondary_j, teacher_n, unruly_j*

Cluster 95: *give_v - gain_v*

Cluster 608: *calculate_v, aerial_n, BDI_np, canvass_v, converge_v, correlation_n, corresponding_j, cumulative_j, deviance_n, deviation_n, eigenvalue_n, eigenvectors_n, estimation_n, Figure_np, fluctuation_n, frequency_n, Ga_np, indicate_v, indicator_n, individualization_n, iteration_n, Lrc_np, marker_n, Mau_np, measurement_n, megabyte_n, mosaicity_n, multiply_v, PDM_np, phi_n, randomly_r, rejection_n, sample_v, sampling_n, scaling_n, scan_n, slice_n, subscales_n, trans_n, variance_n, volume_n, yield_v*

The first conclusion that can be made is that the obtained clusters are not POS coherent. Almost all of the clusters contain words from multiple parts of speech, in many cases it is hard to determine the main part of speech for the cluster. With regard to semantic relatedness, the results are slightly better than

expected. Even at this low corpus size, some semantic relations can be found: Cluster 389(1M-raw) contains words related to health, Clusters 112 (1M-pos), 400(2M-raw), 108(2M-lemma), 700(2M-pos) contain words related to childhood, Cluster 77(2M-raw) contain words related to vacation, Clusters 710(2M-raw), 680(2M-lemma), 608(2M-pos) contain words related to computation. However, even if some words are related, the relationship is generally weak and there are a lot of unrelated words. The results obtained from the 2Ml corpus look better than the results obtained from the 1M, however overall they are not good enough. While the best cluster (112 from 1M pos) is from the POS tagged version of the corpus, at this sample size it is hard to determine whether there is a significant difference between raw, lemmatized and POS tagged.

4.3.2 4M corpus (all words)

The initial expectations for the experiment with 4M corpus were not different from the expectations for the 1M and 2M experiments. The obtained results confirmed the expectations:

Raw text

Cluster 540: *bicycle, 12m, arches, axles, beam, belt, belts, bends, bodywork, boiler, bolted, bolts, booms, boot, boots, brake, brakes, bumpy, cab, cage, carb, carriage, carriageways, collapsed, cupboard, dummy, elevator, FAB, fastened, fences, flames, flaps, flat, fork, gate, gates, gears, hatch, hauled, hauling, headlights, hoist, hooks, hull, idle, jackets, jet, jets, jetty, keys, ladder, lamp, landing, leaking, leaks, lengths, lift, lifts, lights, lock, locking, locks, logs, loosen, markings, mast, mounted, nearside, overhead, overturned, paddle, parked, perimeter, pit, pitched, pod, positioned, pulley, pumped, rack, racks, rad, railing, rails, ramp, refit, rig, roller, rope, rucksack, sails, Scania, screwdriver, seal, seat, shaft, shed, silently, skinned, sockets, steed, stops, stoves, stowed, stretched, tanks, taps, tender, thermostat, throttle, Thunderbird, tighten, tightened, toe, towing, trailing, tunnel, turret, tyre, tyres, underneath, underside, unused, Unusually, veer, wagon, warmers, wheel, wheeled, wheels, Width*

Cluster 271: *child, acknowledging, adolescence, adulthood, bullying, carer, carers, children, couples, disengaged, encouragement, FABE, families, fosters, group, informally, intervention, mothers, mums, parent, parental, parenting, parents, peer, peers, pupil, rhymes, socialise, teenagers*

Cluster 238: *give, advocate, Borders, clarify, complained, dissatisfaction, fears, gave, insisted, neighbours, reacted, reassured, recommending, reconsider, remarked, respond, responded, response, sought, talked, thanked, urged, urging*

Cluster 744: *calculate, accurate, accurately, adhering, adjusting, algorithm, allocations, altering, artificially, assimilation, behavior, bias, calculating, calculation, comparison, complexity, computed, continuous, conversion, differential, discrete, distortion, distributions, dual, element, equations, equilibrium, equivalence, estimate, Examine, experimentation, exponential, FEA, feasible, finite, formula, geometry, indicating, inventory, iteration, linear, measure, measured, mechanism, mesh, method, minimal, mode, modifying, obsolescence, optimal, optimum, parallel, parameters, partial, precision, predicting, proportional, prototype, recoverable, reduces, refined, reversed, reversible, rigid, scaling, shading, shadowing, simplified, simulation, spares, splitting, standardized, static, stochastic, streamlined, structural, structure, system, technique, thermodynamic, timing, uncertainty, variation, weighted*

Lemmatized text

Cluster 115: *bicycle, Audi, BMW, boot, brake, buggy, car, carriage, crash, driver, driving, GTA, headlight, luggage, Mercedes, mileage, minibus, navigator, park, Pinehurst, Porsche, seatbelt, skid, smash, stow, Toyota, truck, unload, vehicle, wheel, wheelchair, windscreen*

Cluster 74: *child, bedtime, flower, nanny, nursery, parent, parental, Readathon*

Cluster 79: *give, billion, compare, factory, household, spend*

Cluster 698: *calculate, 3p, adjust, allele, amplitude, anode, Borehole, brightness, calculation, capacitor, coefficient, compression, compute, conversion, Convert, corresponding, costtoproduce, cyclic, deflection, density, deviation, dial, dilution, entropy, flux, fraction, frequency, gamma, graph, grid, hue, Hue, hz, initialise, interval, iteration, iterative, latitude, longitude, magnitude, marginally, mesh, minus, mosaicity, multiply, offset, outputiterator, parameter, pence, phi, photon, predefined, presets, prevailing, proportional, rectifier, reliably, remapping, sampling, saturation, scaling, separately, sequential, slice, sparse, subhypothesis, trellis, V02max, vectorized, volt, voltage, wavelength, weighted, yield, z, zero*

Lemmatized and POS tagged text

Cluster 690: *bicycle_n, agility_n, airplane_n, alight_r, ATV_np, Audi_np, aviation_n, barge_n, cargo_n, carrier_n, CIRRUS_np, Comet_np, competitively_r, cruiser_n, depot_n, diesel_n, electric_j, handbag_n, hydroelectric_j, liner_n, LNG_np, locomotive_n, LPG_np, mammoth_j, motorise_v, motor_n, onboard_j, overhaul_v, paddle_n, PFA_np, Power_np, power_v, propulsion_n, railroad_n, refuel_v, Rhino_np,*

Rolls_np, Royce_np, sail_n, ship_v, steam_n, Steam_np, tanker_n, tram_n, wagon_n, wheel_v

Cluster 113: *child_n, accomapnied_v, accompany_v, adult_j, adult_n, child_n, equipped_j, recommend_v*

Cluster 75: *give_v, Christmas_np, subject_j*

Cluster 323: *calculate_v, account_v, amount_v, average_n, average_v, calculation_n, changeover_n, commensurate_j, comparator_n, cost_v, deduction_n, deficit_n, double_v, equate_v, estimated_j, estimate_n, estimate_v, euro_n, expenditure_n, fluctuate_v, gram_n, increment_n, inflation_n, maximum_n, multiplier_n, offset_v, outgoings_n, pence_n, pensionable_j, percentage_n, percent_n, quarter_n, rateable_j, roughly_r, RPI_np, statistic_n, sterling_n, total_j, total_v, turnover_n, weighting_n, Yearly_np*

Similar to the experiment with 1M and 2M corpus size, the obtained clusters are not POS coherent. Some of them contain semantically related words: Clusters 271(raw) and 74(lemma) contain words related to childhood, Clusters 744(raw), 698(lemma), 323(pos) contain words related to computation, Clusters 115(lemma), 690(pos) contain words related to vehicles. However, many unrelated words are presented in the clusters as well and the relationship is not strong. The results obtained suggest that the two modified corpora - the lemmatized and the pos tagged could provide better representations.

4.3.3 20M corpus and 40M corpus (all words)

Mikolov et al. [2013] demonstrate results with 25M and 50M corpora. At 25M, they report about 23% accuracy, at 50 - 29%. The maximum accuracy they report at 750M corpus size is 45%. This means that 25M and 50M corpora are enough for respectively 50% and 65% of the best results. Therefore the expectations for the 20M and 40M corpus experiments were to provide more relevant clusters.

20M corpus

Raw text

Cluster 709: *bicycle, 100mph, ABS, adjuster, aero, airbags, axle, axles, belts, bodywork, bogie, bogies, bonnet, brake, brakes, Brakes, braking, Braking, carburettor, clutch, coaster, cog, crank, deflating, dynamo, dyno, fairing, fender, Firebolt, fitted, footrests, freewheel, gear, handbrake, handlebars, headlamp, headlamps, headlights, hitch, immobiliser, inboard, Kangoo, manoeuvrable, mechanic, motorhome, mower, mudguards, nearside, panniers, parachutes,*

pedal, pedals, pillion, Punto, rear, rearward, revving, roller, rotor, saddle, saddles, seat, silencer, speedometer, spoked, spokes, sprocket, steering, stoker, suspension, tacking, tandems, tire, traction, trailer, tricycle, trike, trikes, trundle, tyre, tyres, URoW, warmers, wheel, wheels, windscreen, windshield, wraparound

Cluster 1682: *child, 25s, adoptive, adult, adults, autistic, Bookstart, Camphill, childcare, childminders, children, childs, Complaining, crche, dyslexic, educating, educationally, families, lone, nanny, NCMA, nurseries, parent, parental, Parental, parenthood, parenting, parents, Parents, schooling, socialisation, Sure, truant, YMU*

Cluster 227: *give, able, accomodate, all, always, anybody, anyone, anything, anytime, anywhere, beforehand, begin, best, bring, can, chance, choose, come, decide, do, doing, done, Either, else, every, everybody, everyone, Everyone, everything, expect, feel, find, forget, get, go, happen, happening, happens, happy, hard, have, hold, hope, Hopefully, how, if, If, know, leave, let, Let, lets, letting, like, look, make, need, our, ours, personally, prefer, put, quickest, Remember, remembering, remind, reminding, right, see, So, somebody, someone, something, stays, straightaway, suits, sure, take, tell, tempted, them, things, to, touch, try, unsure, us, want, wanting, wants, we, We, what, whatever, Whatever, whenever, Whenever, wherever, whoever, why, willing, wish, wo, workmates, worry, you, You, youll, your, youre, yours, yourself, yourselves*

Cluster 836: *calculate, assuming, Assuming, calculate, calculated, calculating, calculation, calculations, formula, Suppose*

Lemmatized text

Cluster 37: *bicycle, Bicycle, bike, Bike, biker, Bikes, BMX, C5, chopper, Cycles, Cyclescheme, Helmet, kart, moped, motorbike, motorcycle, Motorcycle, MTB, pannier, quad, ride, Ride, skateboard, triathalon, tricycle, trike*

Cluster 123: *child, adoptive, adult, adulthood, age, Cafcass, Camphill, carer, caring, childhood, children, childs, crche, educate, EHE, family, grandchild, grandparent, home, kindergarten, lone, nursery, parent, parenthood, parents, playgroup, playgroups, playschemes, postive, separated, sibling, socialisation, ss, upbringing, YMU, young*

Cluster 1460: *give, a, able, about, all, already, are, as, be, best, better, can, choice, choose, constantly, deal, different, differently, easier, expect, explain, good, have, help, how, idea, importantly, make, many, Many, mean, might, more, need, our, own, people, pressurise, Px, real, responsibly, sensibly, simply, solve, some, that, their, them, themselves, There, they, this, those, to, understand, want, way, we, well, what, What, whatever, where, whole, will, willing, with, would*

Cluster 1007: *calculate, adjust, adjustment, alter, appreciable, attrition, calculated, CF, change, compensate, constant, continuously, convergence, costtoproduce, cumulative, decrease, difference, discounting, duration, effect, elasticity, exogenous, factor, fluctuate, fluctuation, incrementally, intensity, KLS, lessor, likelihood, magnitude, measure, minimize, miniscule, offset, overall, OwnCal, predetermine, residual, result, resultant, resuspension, RoL, saturation, scale, sensitivity, shift, substitutability, timing, triviality, uncertainty, value, variation, vary*

Lemmatized and POS tagged text

Cluster 571: *bicycle_n, Emma_np, Restall_np*

Cluster 694: *child_n, adolescence_n, adolescent_n, adulthood_n, attitude_n, autism_n, autistic_j, bodily_j, burnout_n, Camphill_np, caring_n, childhood_n, emotional_j, emotionally_r, empathetic_j, empathy_n, esteem_n, estrangement_n, family_n, fatherhood_n, fraternal_j, infancy_n, Korczak_np, mentally_r, parent_n, parent_v, personality_n, postive_j, promoting_j, psychologically_r, puberty_n, resourcefulness_n, schoolwork_n, sibling_n, socialisation_n, socialise_v, teenage_j, wellbeing_n, YMU_np, young_j*

Cluster 245: *give_v, able_j, already_r, and_o, a_o, as_s, aware_j, benefit_v, chairperson_n, chairpersons_n, commit_v, confidence_n, confident_j, continue_v, effort_n, endeavour_v, expect_v, financially_r, for_s, forward_r, fulfil_v, future_n, helpfulness_n, help_v, hope_v, important_j, incentivising_v, interviewee_n, keen_j, make_v, many_j, Many_j, motivator_n, other_n, our_o, own_j, people_n, positive_j, positively_r, prepare_v, productively_r, reassurance_n, responsibly_r, seek_v, sensibly_r, supportive_j, their_o, themselves_o, those_o, to_o, vitally_r, whilst_s, who_o, willing_j, willingness_n, with_s, work_v*

Cluster 675: *calculate_v, accurately_r, aggregate_j, aggregate_v, arbitrage_n, baseline_j, baseline_n, calculated_j, categorisation_n, cluster_v, comparison_n, complementarity_n, converge_v, correlation_n, cumulative_j, dataset_n, datasets_n, datum_n, define_n, derive_v, deviation_n, DF_np, disaggregate_v, dispersion_n, distribution_n, diurnal_j, DMRB_np, elasticity_n, estimate_n, estimation_n, exogenous_j, extrapolate_v, extrapolation_n, generate_v, graph_v, gravimetric_j, incrementally_r, indicated_j, indicate_v, indicator_n, indices_n, ir_n, iteration_n, KLS_np, measurement_n, measure_v, medium_j, minima_n, optimal_j, phase_n, pool_v, practicability_n, predetermine_v, prediction_n, predict_v, produce_v, projection_n, Q2_np, QES_np, quantify_v, realisable_j, relative_j, reliably_r, resultant_j, result_n, resuspension_n, RMS_np, robustness_n, RoL_np, sample_n, sample_v, sampling_n, scale_n, scale_v, scenario_n, TEOM_np, throughput_n, unambiguously_r, utilisation_n, variability_n, variation_n, weighting_n*

40M corpus

Raw text

Cluster 2344: *bicycle, 100mph, 4WD, 4x4, 4X4, APB, bicycles, Bicycles, bike, biker, bikers, bikes, Brompton, buggies, buggy, cabs, car, diggers, dismount, drive, dumper, dumpers, forklift, forklifts, garage, Harleys, haulage, HGV, kart, karts, lawnmower, lorries, lorry, mechanic, minibuses, Moffett, moped, mopeds, MoT, MOT, motor, motorbike, motorbikes, motorcycle, motorcycles, motorhome, motorised, offroad, panniers, piggyback, racetrack, recumbent, recumbents, rickshaw, ride, roadwork, roadworthiness, roadworthy, roughest, scooter, scooters, Scrutineering, sidecar, skateboard, skateboards, Snob, strapping, sweepers, tachographs, tandems, towing, trackside, tractor, tractors, trailer, trailers, tricycle, tricycles, trike, trikes, truck, trucks, trundle, unladen, URoW, vehicle, vehicles*

Cluster 2086: *child, ACeBabes, ACeKids, babysitters, caring, childcare, Childcare, childcarer, childcarers, childminder, childminders, Childminders, childminding, childproofing, Childworks, creches, dads, EYDCP, grandchild, grandparent, juggled, mums, nannies, Nannies, nanny, Nanny, nurseries, nursery, parent, parenting, playgroup, playscheme, threes*

Cluster 1971: *give, able, all, are, best, bring, can, carefully, choice, choose, choosing, chose, decide, Either, Failing, find, finding, follow, giving, help, how, importantly, letting, make, making, need, offer, our, own, prefer, recommend, suits, take, their, to, unsure, We, whatever, Whatever, whenever, wherever, Wherever, Whether, Whichever, willing, with, your*

Cluster 100: *calculate, assessable, assuming, Assuming, calculated, calculates, calculating, calculation, closest, count, counting, counts, cumulatively, decimal, decimals, denominator, detours, divide, dividing, FHR, formula, fractions, HRmax, ie, misclassification, multiples, multiplied, multiplier, multipliers, multiply, Multiply, multiplying, points, poundage, priors, probabilities, probability, recalculated, shortest, subtract, subtracted, subtracting, summing, Suppose, z, zero*

Lemmatized text

Cluster 289: *bicycle, airnimal, apb, APB, bicycle, Bicycle, bicyclist, bike, Bike, Bikers, Bikes, Birdy, BMF, BMX, brompton, Brompton, Bromptons, cyclefest, CycleFest, Cycles, Dahon, dosh, fatboy, Folding, FSN, handlebars, helmeted, Izzi, Juden, lube, MCN, motorbike, motorcycle, Motorcycling, Moulton, Moultons, MTB, NABD, offroad, Organics, Packhorse, pannier, pedalling,*

Pilling, recumbent, recumbents, ride, Snob, triathalon, tricycle, trike, Trike, Trikes, Wheelers

Cluster 378: *child, adult, age, babysitter, Bhandari, Camphill, childminder, childminders, Childminders, childminding, Childminding, children, creche, daycare, educate, Gervasio, grandparent, intergenerational, Jetha, kindergarten, Kindermusik, Kiya, Nalyaka, NCMA, nursery, orphanage, Orphanage, parent, playgroup, playgroups, playscheme, playschemes, playworkers, Portage, respite, schoolchildren, schoolwork, socialisation, socialise, teenager, YMU, young, YOUNGOFF, yrs*

Cluster 164: *give, a, able, all, allow, already, also, are, as, aware, be, behave, best, can, cannot, carefully, certain, choose, clear, constantly, deal, different, differently, dislikes, either, expect, explain, firstly, for, have, how, however, idea, importantly, in, instead, interviewee, make, Many, may, mean, might, more, need, no, not, only, other, our, own, people, point, possible, prefer, properly, reassurance, regardless, respond, same, sensible, simply, some, step, such, take, that, the, their, them, themselves, There, therefore, these, they, this, those, to, treat, understand, usually, way, we, whatever, where, wherever, whether, which, whilst, whole, will, willing, with, without, worthwhile, would*

Cluster 2406: *calculate, actual, adjusted, aggregate, calculated, calculation, cashable, comparator, compare, conventionally, cumulative, cumulatively, equalisation, equate, estimate, estimated, ets, FDR, figure, incremental, LFEPA, maintainable, marginal, marginally, Maserati, multiplier, nominal, NPV, offset, outperform, overheads, PAV, percentage, QALY, quantifiable, recalculate, RPI, simplification, threshold, throughput, TMV, total, unmeasured, unrecoverable, value, weighted*

Lemmatized and POS tagged text

Cluster 912: *bicycle_n, ceremonial_j, donor_n*

Cluster 1114: *child_n, adolescent_n, adulthood_n, adult_j, adult_n, age_j, age_n, age_v, Bhandari_np, bully_v, Camphill_np, commited_v, disaffected_j, educate_v, Fabe_np, intergenerational_j, Jetha_np, kindergarten_n, leaver_n, mainstream_j, Montessori_np, old_n, pairing_n, preschool_j, Prus_np, schoolchildren_n, schooling_n, school_n, schools_j, school_v, schoolwork_n, socialisation_n, truancy_n, truant_j, truant_v, underachieve_v, under_r, unruly_j, YMU_np, young_j, youngster_n*

Cluster 113: *give_v, able_j, all_o, asking_n, ask_v, begrudge_v, can_m, carefully_r, decide_v, every_o, fill_v, find_v, fob_v, how_o, if_s, keep_v, make_v, meantime_n, more_j, need_m, need_v, once_s, option_j, our_o, out_r, promptly_r, Px_np, reapply_v, Should_m, sure_r, take_v, them_o, then_r, they_o, this_r, time_n,*

to_o, unable_j, unsure_j, us_o, we_o, whenever_o, Whenever_o, will_m, wish_v, would_m, you_o, your_o, yourself_o

Cluster 890: *calculate_v, above_np, accurately_r, approximate_v, calculated_j, calculate_v, calibrate_v, calibration_n, chart_n, chisquare_n, counting_n, covariates_n, delta_j, deviation_n, eigenmode_n, extrapolate_v, graph_n, graph_v, histogram_n, hydrograph_n, lad_j, mean_j, measured_j, measurement_n, measure_v, minima_n, misclassification_n, nomogram_n, numerical_j, observed_j, oscillatory_j, Ped_np, plot_v, Poisson_np, prediction_n, predict_v, randomly_r, recalculate_v, repeatable_j, residual_n, result_n, Rummy_np, scatter_n, scatterplot_n, seismologist_n, show_v, t0_n, tabulate_v, timescale_j, typical_j, unrecord_v, variance_n, xy_j*

The results from these experiments did not meet the initial expectations. While the clusters are slightly better than the ones obtained from the experiments with 1, 2 and 4 million corpus, they are still not clearly good. The clusters are still mostly not POS coherent. There is some semantic relatedness in most of them, however there are still a lot of unrelated words and in many clusters the relationship is hard to find. The results obtained from 40M corpus are better than the results from the 20M, but not by a huge margin. However, the results from these experiments confirmed the expectations about the processing of the corpus: the clusters obtained from the lemmatized corpus and from the lemmatized and pos tagged corpus are better than the results obtained from the raw corpus. It looks like the lemmatized version provides highest cluster quality in this experiment.

4.3.4 1M and 2M corpuses (10 000 words)

The clusters from 1M experiment are identical to the ones from the first set of experiments. The obtained results for 2M were the following:

Raw text

The token "bicycle" did not appear at least 8 times in the corpus

Cluster 400: *child, adolescent, allegations, Ayurveda, basics, childhood, Developmental, Early, enjoyed, enrich, esteem, outdoors, PADI, preschool, teaches, trainer*

Cluster 244: *give, gather, lots*

Cluster 360: *calculate, 7, aids, LORA, Measure, minimal, precise, quantity, recommended, spares, vs*

Lemmatized text

Cluster 788: *bicycle, accomodation, apartment, atmosphere, cater, catering, charming, comfort, comfortable, convenient, cosy, craft, friendly, guest, holiday, hotel, indulge, lively, luxury, memorable, overnight, reception, relax, renovate, restaurant, sailing, showroom, spacious, studio, stunning, stylish, superb, surroundings, wedding*

Cluster 108: *child, deprive, enjoyable, Further, retirement, select*

Cluster 82: *give, HIV, immune, misuse, nervous, population*

Cluster 713: *calculate, accurate, accurately, algorithm, calculation, checking, chromosome, circadian, coherent, component, compress, compression, computational, compute, construct, continuous, contour, detector, discrete, Element, encode, entropy, equation, estimation, experimental, finite, foetal, formula, functional, Ga, gat, genetic, genome, geometry, graph, histogram, inference, input, interaction, logical, manipulate, mapping, matching, mathematical, matrix, measurement, mesh, method, model, multiply, mutation, node, numerical, optimal, optimum, parallel, parameter, pattern, PDM, physiological, predictor, processing, randomly, reconstruct, reconstruction, relational, reproduction, rhythm, rotation, sampling, scan, scenario, segment, segmentation, shading, shape, simulation, simultaneous, spatial, stated, suitability, textbook, ubiquitous, ultrasound, variation, vertical, volume*

Lemmatized and POS tagged text

Cluster 338: *bicycle_n, assemble_v, awesome_j, basin_n, bell_n, blue_n, bronze_n, cake_n, carve_v, cassette_n, colourful_j, crystal_n, deck_n, drawer_n, drill_v, elephant_n, finish_n, flash_v, fly_n, fold_v, funky_j, gear_n, glitter_n, glitter_v, gold_n, hang_v, hip_n, jar_n, jewellery_n, laughter_n, leather_n, magically_r, medal_n, mini_j, nicely_r, occasional_j, orange_n, pack_v, paint_v, pine_n, pink_j, rubber_n, seal_n, shelf_n, short_n, silver_j, soap_n, spider_n, super_j, trouser_n, truck_n, twist_v, vintage_j, yarn_n*

Cluster 105: *child_n*

Cluster 80: *give_v, buffer_n, byte_n, record_v, retrieve_v, track_n*

Cluster 252: *calculate_v, calculate_v, desire_v, exceed_v, fixed_j, ie_n, ingest_v, maximum_j, maximum_n, minimal_j, minimum_j, minimum_n, non_j, preceding_j, required_j, Secure_np, specified_j, temper_n, Tenancy_np*

Overall, the results from these experiments were not good.The clusters are not POS coherent and most of the clusters consist of unrelated words. There are few clusters with some related words: Cluster 400 (raw) has some related words

("child", "childhood", "adolescent", "preschool"), Cluster 788 (lemma) contains many words related to vacation, Cluster 713 (lemma) contains many words related to calculation. Even in the clusters that have some related words, there are a lot of unrelated ones. These experiments also confirm the expectation that 1M and 2M corpus size is too low to generate relevant vector representations and clusters. At least in the analyzed cases, it seems that lemmatized corpus provides the best results so far.

4.3.5 4M corpus (10 000 words)

The obtained results for 4M were the following:

Raw text

Cluster 717: *bicycle, cases, competitor, continues, ship*
Cluster 561: *child, arm, arms, joints, limbs, muscles*
Cluster 238: *give, afforded, cite, entertain, giving, lend, offering*
Cluster 1082: *calculate, acquire, adherence, adopt, analyse, analyze, anticipate, assess, broaden, consider, construct, convey, define, demonstrate, describe, determine, devise, emphasise, encompass, engage, establish, evaluate, examine, execute, exploit, extend, facilitate, fulfil, identify, incorporate, inform, integrate, interpret, introduce, investigate, minimise, modify, prioritise, pursue, recognising, refine, reflect, reinforce, relate, reproduce, satisfy, tackle, transform, undermine*

Lemmatized text

Cluster 715: *bicycle, accommodate, adjustable, baggage, Battery, belt, bike, Binz, cabin, caliper, car, carriage, chassis, diesel, economical, electric, engine, generator, hydraulic, lightweight, luggage, motor, motorcycle, parking, piston, racing, Rear, rider, safely, spare, suspension, truck, tyre, unusually, vehicle*
Cluster 113: *child, anywhere, police*
Cluster 301: *give, amaze, apologise, appreciate, ask, assure, beg, confess, convince, hesitate, instruct, personally, persuade, reassure, remind, said, suppose, suspect, tell, wary, whenever*
Cluster 316: *calculate, accuracy, accurately, algorithm, analysis, behavior, calculation, categorization, CF, Characteristic, classification, comparison, complexity, compute, correlation, cumulative, Defining, derivative, deviation, DF, differential, diffusion, dimension, equilibrium, estimation, example, experiment, Feature, Figure, formula, formulation, frequency, geometric, hypothesis,*

instance, linear, measure, model, observed, optimal, parameter, partial, pattern, prediction, probability, prototype, Prototype, recreate, regression, sample, sampling, scale, scenario, sensitivity, similarity, threshold, timescale, uncertainty, value, variability, variable, variance, variant, variation, yield, zero

Lemmatized and POS tagged text

Cluster 437: *bicycle_n, appliance_n, bike_n, Binz_np, boiler_n, carrier_n, control_n, dedicated_j, electrical_j, electric_j, engine_n, generator_n, machinery_n, motorcycle_n, motor_n, operator_n, power_v, quad_n, repair_n, repair_v, service_v, ship_v, transportation_n, truck_n, tyre_n*

Cluster 92: *child_n, activity_n, adolescent_j, adolescent_n, adult_n, age_n, athlete_n, developmental_j, learner_n, lesson_n, maturity_n, nappy_n, parental_j, pupil_n*

Cluster 75: *give_v, await_v, Christmas_np, subject_j*

Cluster 316: *calculate_v, actual_j, allocation_n, availability_n, duration_n, fixed_j, guaranteed_j, limited_j, maximum_n, minimal_j, minimum_j, minimum_n, recommended_j, recovery_n, reduced_j, scheduled_j, spare_n, specified_j*

This set of experiments provides much better results compared to the experiment with 2M and compared to the initial expectations. In many clusters there is significant improvement in the POS coherence. The clusters are still not perfect, but in many cases the majority of the words are from the same POS. There is also a noticeable improvement of the semantic relatedness: Cluster 561 (raw) contains body parts; Cluster 238 (raw) contains "giving", "lend", "offering", "give"; Clusters 715 (lemma) and 437(pos) contain multiple vehicles; Cluster 301 (lemma) contains multiple verbs for talking and persuading; Cluster 316 (lemma) contains nouns related to maths; Cluster 92 (pos) contains age-related person nouns; Cluster 316 (pos) contains nouns and adjectives related to duration. While in many cases the relation is not too strong, it is clearly presented in multiple clusters. The conclusion from this experiment is that at a corpus size of 4M, with using only the 10 000 most frequent lemmas, Word2Vec is able to catch some relevant semantic information. It is also clear that in the discussed examples, lemmatized and POS tagged corpus provides better vector representations and clusters.

4.3.6 20M and 40M corpus (10 000 words)

The obtained results for 20M and 40M were the following:

20M corpus

Raw text

Cluster 860: *bicycle, accessories, adaptations, clothing, equipment, footwear, helmet, helmets, protective*

Cluster 551: *child, baby, breastfeeding, carer, childcare, childhood, counselling, guardian, infant, lone, maternity, midwife, nanny, parent, parental, parenting, paternity, pregnancy, pregnant, teenage*

Cluster 24: *give, begin, borrow, bring, catch, come, fill, find, follow, get, go, happen, hold, join, kill, leave, listen, live, lose, make, miss, move, pass, pick, push, reach, repeat, save, see, show, speak, stand, start, stay, stop, take, watch*

Cluster 859: *calculate, assign, attach, compile, execute, handle, load, manipulate, modify, obtain, verify, without*

Lemmatized text

Cluster 106: *bicycle, car, commuter, cyclist, drive, driver, driving, lane, motor, motorcycle, motorist, parking, pavement, pedestrian, road, roadside, scooter, toll, vehicle, wheelchair*

Cluster 123: *child, adult, camping, childrens, gym, indoor, kid, nursery, outdoor, outdoors, parent, playground, swimming, toddler*

Cluster 75 : *give, add, broaden, find, gain, intended, point, reach, spell, widen*

Cluster 750: *calculate, accord, average, comparable, compare, count, cumulative, equate, equivalent, estimate, estimated, exceed, percent, percentage, respectively, total*

Lemmatized and POS tagged text

Cluster 41: *bicycle_n, adjustable_j, apparatus_n, bearing_n, bike_n, brake_n, circuit_n, cylinder_n, dual_j, engine_n, exhaust_n, gear_n, generator_n, harness_n, hose_n, jet_n, loading_n, mast_n, motorcycle_n, motor_n, mount_v, piston_n, power_v, pump_n, rear_j, rocket_n, roller_n, steering_n, suspension_n, trailer_n, tyre_n, valve_n, ventilation_n*

Cluster 488: *child_n, adult_n, carer_n, carers_n, caring_j, childcare_n, deaf_j, elderly_j, hospice_n, infant_n, lone_j, nanny_n, parental_j, parent_n, parent_v, toddler_n, young_j*

Cluster 231: *give_v, collect_v, count_v, deposit_v, exchange_v, hold_v, keep_v, meet_v, obtain_v, place_v, process_v, receive_v, retain_v, serve_v, transfer_v*

Cluster 612: *calculate_v, achieve_v, assess_v, compensate_v, control_v, correct_v, desire_v, detect_v, determine_v, eliminate_v, generate_v, identify_v, measure_v, minimise_v, monitor_v, optimal_j, optimise_v, precise_j, predict_v, quantify_v, reduce_v, refine_v, test_v, yield_v*

40M corpus

Raw text

Cluster 1103: *bicycle, bike, bikes, BMW, Bosch, cab, car, carriage, cars, diesel, driver, driving, Leyland, lorry, Mercedes, MG, motor, motorcycle, Renault, rider, riding, Rover, trailer, trailers, truck, trucks, van, vans*

Cluster 908: *child, carer, guardian, parent, parental, spouse*

Cluster 728 : *give, abandon, accept, collect, commit, convince, deny, disclose, dismiss, hide, hold, ignore, impress, justify, kill, lose, make, persuade, prove, reject, resist, tolerate*

Cluster 705: *calculate, analyse, ascertain, assess, clarify, compare, construct, define, describe, detect, determine, establish, evaluate, examine, explain, identify, illustrate, inform, interpret, investigate, locate, obtain, predict, refine, simplify*

Lemmatized text

Cluster 34: *bicycle, bike, brake, cab, carriage, chassis, clutch, drive, fork, load, loading, lorry, motor, motorcycle, pedal, rack, rear, roller, saddle, seat, suspension, tow, tractor, trailer, truck, tyre, wheel*

Cluster 123: *child, carer, carers, family, foster, grandparent, guardian, lone, parent, parental, relative*

Cluster 241 : *give, accord, allow, lend, owe, remind*

Cluster 6: *calculate, accurately, actual, approximate, baseline, calculation, calibration, classification, compute, datum, differential, distribution, equation, estimation, exact, formula, indicator, linear, matrix, measure, measurement, metric, multiply, numerical, optimal, optimum, partial, periodic, precise, predict, prediction, probability, proportional, quantify, random, randomly, regression, residual, respectively, sample, sampling, statistical, systematically, threshold, variability, variance, zero*

Lemmatized and POS tagged text

Cluster 363: *bicycle_n, brake_n, bus_n, cab_n, cargo_n, car_n, carriage_n, chassis_n, coach_n, depot_n, driver_n, freight_n, locomotive_n, lorry_n, motorcycle_n, motor_n, operator_n, passenger_n, rail_n, railway_n, steering_n, taxi_n, tractor_n, trailer_n, train_n, truck_n, van_n, vehicle_n, wagon_n*

Cluster 121: *child_n, adult_n, age_n, baby_n, birth_n, breastfeed_v, infant_n, lone_j, parent_n, parent_v, pregnancy_n, pregnant_j, teenager_n, toddler_n*

Cluster 682: *give_v, acknowledge_v, address_v, ascertain_v, assess_v, clarify_v, conclude_v, confirm_v, consider_v, define_v, demonstrate_v, describe_v, determine_v, discuss_v, document_v, emphasise_v, emphasize_v, envisage_v, examine_v, explain_v, formulate_v, highlight_v, identify_v, illustrate_v, indicate_v, inform_v, interpret_v, investigate_v, note_v, outline_v, predict_v, present_v, question_v, recognise_v, reflect_v, report_v, reveal_v, revisit_v, show_v, state_v, stress_v, structure_v, suggest_v, summarise_v, understand_v*

Cluster 524: *calculate_v, absolute_j, ie_n, maximum_j, measure_n, measure_v, minimise_v, minimize_v, multiply_v, practicable_j, quantify_v*

There are several conclusions that can be made from these clusters. Overall, the results are better than the results obtained from the 4M corpus - the majority of the clusters are POS coherent and in most of the cases there is at least some kind of semantic relationship. The results obtained from 40M corpus are better than the results obtained from 20M corpus. In the analyzed clusters, the lemmatized and POS tagged corpus provides the best results, while the raw text corpus provides the worst results. It is also evident that the quality of the noun clusters is higher than the quality of the verb clusters. The results of this set of experiments confirmed the expectations - even tho there are some unrelated words and some bad clusters, the overall quality of the obtained results indicates that the vector representations and the clusters represent relevant semantic features of the words.

4.3.7 Summary

Overall, the different sets of experiments provided a lot of valuable data for analysis. The set of experiments that used all of the words did not provide very good results. For all of the experiments, the clusters were are POS coherent. With regard to the semantic relatedness of the words in the corpus, the results are mixed. Many of the obtained clusters contain some semantic words, but the relation is not always clear and there are often many unrelated words. Increasing the corpus size led to an improvement of the results, however even at 40 million token size,

the good clusters are still relatively low number. The clusters containing mostly nouns are better in general than the clusters containing mostly verbs.

The set of experiments using only the 10 000 most frequent words provided relatively good results. At a corpus size of 4M, most of the clusters have a dominating POS tag and some of them have at least some degree of semantic relatedness. At a corpus size of 20M the POS coherence is evident and the semantic relation is stronger. With regard to both POS coherence and semantic similarity, the best results are obtained using 40M corpus. In this set of examples, increasing the corpus size clearly improves the quality of the results. Once again, clusters containing mostly nouns are better than the clusters containing mostly verbs.

In both sets of experiments, the control test using Skip-gram learning algorithm on the 40M corpus did not provide results that were better than results obtained by CBOW. In both sets of experiments, the linguistic preprocessing on the corpus (lemmatization and pos tagging) demonstrated improvement of the results.

4.4 Comparing the process and the obtained clusters

One of the goals of this thesis is to compare the process of vector generation and the quality of the obtained results using two different methodologies. The setup of the experiments was prepared with that goal in mind - the experiments were run on the same computer, using the exactly same corpora. Clusters containing the same word were chosen and analyzed to demonstrate the quality of the obtained vectors and clusters of vectors. The obtained data allowed to make comparisons in three directions - computational complexity and time required for the process; overall quality of the obtained vectors and clusters; and the effect of increasing corpus size.

With respect to computational performance, all tests were run on a single core 64 bit computer. Both methodologies have a way of using multiple cores, so the process can be sped up. The timing of the different experiments is described in table 4.2.

Methodology	1M	2M	4M	20M	40M
CBOW (10k)	6 min	12 min	25 min	125 min	250 min
CBOW (all)	6 min	15 min	31 min	155 min	310 min
Skip-Gram	n/a	n/a	n/a	n/a	1000 min
CLUTO	12 min	25 min	50 min	330 min	550 min

Table 4.2: Execution times for different experiments

From the point of view of computational time, CBOW provides the best results. This was expected as one of the main goals presented by Mikolov et al. [2013] is to create algorithm with relatively low computational complexity, which can be applied to corpora of size above 1B words. CLUTO is slower than CBOW, but faster than Skip-gram. It is interesting to note that increasing the number of lemmas does not lead to significant increase in the computational time for CBOW. The computational time of my own python scripts that preformat the corpus is included in the total - preparing a 40M corpus for Marti et al. [2015] takes about 1 hour, preparing a 40M corpus for Mikolov - about 5 min.

With respect to the POS coherence, Marti et al. [2015] provides better results than both CBOW and Skip-Gram. For all of the experiments, and especially with corpus size of 4 mil and above, the clusters obtained by CLUTO were mostly POS coherent. Word2Vec demonstrated relative POS coherence only at corpus size of 20M and higher and only when using the most frequent 10 000 words. At 40M corpus size, the clusters contained some words of different POS, but the results were overally good. No significant difference between CBOW and Skip-gram algorithms was notices. When using all of the words Word2Vec did not manage to obtain POS coherent clusters.

With respect to the semantic relatedness, both CBOW and CLUTO provide poor results at corpus size of 1M and 2M. At corpus size 4M and 20M CLUTO provides better results as CBOW (10k) still has a lot of unrelated words in the clusters and CBOW (all) obtains very few good clusters. At 40M tokens, both CLUTO and CBOW (10k) provide good results, while CBOW (all) does not. The clusters obtained by CLUTO still look better, however the quality of the clusters obtained by CBOW (all) is comparable. Both methodologies obtain better results in noun clusters than in verb clusters.

With respect to the effect of increasing corpus size, the methodology of Marti et al. [2015] demonstrates significant improvement with increasing the corpus size from 1 to 20M, but the improvement between 20 and 40M is not that relevant. CBOW (10k) demonstrates steady improvement with increasing the corpus size from 1 to 40M. Using the 10 000 most. CBOW(all) demonstrates some improvement, however the quality of the clusters remains bad.

In summary the clusters obtained by the methodology suggested by Marti et al. [2015] are better than the clusters obtained by Word2Vec using the CBOW learning algorithm at least at the corpus size between 1 and 40 mil tokens. Limiting the number of words that CBOW works with does seem to obtain decent results at a corpus size of at least 40M. On the computational complexity side, CBOW is performing much better on larger corpora, taking much less time to obtain the vectors.

Chapter 5

Conclusions

5.1 Overview

This thesis focuses on automatic unsupervised methods for obtaining clusters of semantically related words from a corpus as a way for representing their meaning. It describes the process of applying two different methodologies and compares the results. The first method, suggested by Marti et al. [2015] has not been used for English until now. The second method is well-known and applied for English, however the thesis suggests modifications that can lead to improvement of the results.

The replication of the methodology, suggested by Marti et al. [2015] included the choice of the corpus, the modification of the corpus - both formally and linguistically, and five different experiments with different corpus sizes. The analysis of the results determined that the methodology is applicable for English and can provide relevant POS coherent clusters of semantically related words. The minimum corpus size for obtaining relevant semantic information was found to be 4M tokens. At a corpus size of 20 mil tokens, significant number of good clusters were obtained. The methodology seemed to work better when clustering nouns than when clustering verbs. Nevertheless many good verb clusters were obtained as well.

The usage of Word2Vec to obtain clusters also provided good results, but only after modifying the original process by increasing the occurrence threshold and adding linguistic preprocessing. Using the original Word2Vec methodology provided low quality clusters. The minimum corpus size for obtaining some relevant semantic information was found to be 4M tokens and the first overall good results were obtained at 40M corpus. The obtained clusters were POS coherent after corpus size of 20M. The results showed significant improvement with increasing the corpus size. Similar to the methodology of Marti et al. [2015], the clusters, which

contained mostly nouns were relatively better than the clusters, which contained mostly verbs.

Apart from the quality of the obtained clusters, the two methodologies were compared in at least two aspects: corpus pre-requirements and computational complexity. Word2Vec is designed to work with raw corpora, without any annotation or linguistic pre-processing. The results obtained in this thesis suggest that replacing the word token by the corresponding lemma or by the lemma and the POS tag can lead to improvement of the quality of the clusters.

The methodology suggested by Marti et al. [2015] requires the corpus to be lemmatized, POS tagged and syntactically parsed. The syntactic parsing is one of the more complex pre-processing operation and as Word2Vec makes no use of it, its requirements for the corpus are easier to meet.

With respect to computational complexity and time, Word2Vec (the CBOW algorithm in particular) performs better than CLUTO. The lower pre-requirements and the better computational performance suggest that Word2Vec (CBOW) will require less computational time to work with corpora of bigger size and will eventually obtain better results than the ones demonstrated in these experiments.

5.2 Contributions of the thesis

The contributions of this thesis can be summarized in several directions:

1. I successfully replicated Marti et al. [2015] for English, in the part related to vector generation and clusterization. I verified the applicability of the methodology for a second language and made it available for future research.

2. I created a program to convert a standardized POS tagged and syntactically parsed English corpus to the format required by the methodology of Marti et al. [2015]

3. I compared the methodologies of Marti et al. [2015] and Word2Vec in terms of the computational requirements, the corpus pre-processing requirements, the corpus size requirements, and the quality of the obtained clusters. The obtained data can be used for future research and reference.

4. I suggested modifications of the corpus and the parameters that can improve the results of Word2Vec. The suggestions were confirmed by the experimental results.

5.3 Future work

This thesis opens several lines of future research. First, the process of Marti et al. [2015] can be replicated completely, up to the point where candidates to be constructions are generated. Second, the performance of CLUTO can be tested with higher number of words and evaluated both computationally and on the quality of the clusters. Third, the two methodologies can be compared using corpora of bigger size: 100M, 200M, 500M or on a different language. Fourth, Word2Vec can be tested with corpora of bigger size (500M+) to confirm or deny the suggested hypothesis that lemmatization and/or POS tags improve the vector quality. Finally, the role of the number of words and clusters for Word2Vec can be evaluated using different mathematical and statistical tools.

Bibliography

Antti Arppe, Gaetanelle Gilquin, Dylan Glynn, Martin Hilpert, and Arne Zeschel.
Cognitive corpus linguistics: five points of debate on current theory and methodology.
Corpora, 5(1):1–27, 2010.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta.
The wacky wide web: A collection of very large linguistically processed web-crawled corpora.
Language Resources and Evaluation, 43(3):209–226, 2009.

Marco Baroni and Alessandro Lenci.
Distributional memory: A general framework for corpus-based semantics.
Comput. Linguist, 36(4):663–721, 2010.

Gemma Boleda and Katrin Erk.
Distributional semantic features as semantic primitives or not, 2015.

William Croft and D. Alan Cruse.
Cognitive Linguistics.
Cambridge University Press, 2004.

Katrin Erk.
Vector space models of word meaning and phrase meaning: A survey, 2012.

Stefan Evert.
Corpora and collocations, 2008.

J Fodor, M. F. Garret, Walker E. C., and Parkes C. H.
Against definitions.
Cognition, 8(3):263–367, 1980.

Zelling Harris.
Distributional structure.
Word, 10(23):146–162, 1954.

R. S. Jackendoff.
Semantic Structures.
The MIT Press, 1990.

Richard Johansson.
Dependency syntax in the conll shared task 2008, 2007.

George Karypis.
Cluto a clustering toolkit., 2003.

Alessandro Lenci.
Distributional semantics in linguistic and cognitive research.
Rivista di Linguistica, 20(1):1–31, 2008.

M. Antonia Marti, Manuel Bertran, Mariona Taule, and Maria Salamo.
Distributional approach based on syntactic dependencies for discovering constructions, 2015.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space, 2013.

Hermann Moisl.
Cluster Analisys for Corpus Linguistics.
De Gruyter Mouton, 2015.

Pavel Pecina.
Lexical association measures and collocation extraction.
Language resources and Evaluation, 44:137–158, 2010.

Michael Tomasello.
First steps toward a usage-based theory of language acquisition.
Cognitive Linguistics, 11(1-2):61–82, 2000.

Peter D. Turney and Patrick Pantel.
From frequency to meaning: Vector space models of semantics.
J. Artif. Int. Res., 37(1):141–188, 2010.

T Winograd.
On primitives, prototypes, and other semantic anomalies, 1978.