tions.

Mikolov et al. [2013a] suggest a different approach towards extracting vector representations and grouping. Their methodology is based on deep learning and is intended for quick processing of very large corpora. Word2Vec[1], the tool they present, includes an integrated algorithm for grouping words based on proximity in space. The context they use for vector extraction is simple co-occurrence within a specified window of tokens. Originally, they make no use of linguistic preprocessing such as lemmatization, part of speech tagging or syntactic tagging. As part of this paper we evaluate the effect of linguistic preprocessing on the obtained vectors and groups.

2.3 Data and Tools

In this section we present the corpus that we use in the evaluation (Section 2.3.1) and the two methodologies (Section 2.3.2 and Section 2.3.3).

2.3.1 The Corpus

For all of the experiments described in this paper, we use PukWaC [Baroni et al., 2009][2]. It is a 2 billion word corpus of English, built up from sites in the .uk domain. It is available online and is already preprocessed: XML tags and other non-linguistic information have been removed, it is lemmatized, PoS tagged and syntactically parsed. The PoS tagset is an extended version of the Penn Treebank tagset. The syntactic dependencies follow the CONLL-2008 shared task format.

2.3.2 Grouping with CLUTO

DISCOveR [Martí et al., 2019] is a methodology for identifying candidates to be construction from a corpus. It uses vector representations, extracted from a corpus. CLUTO [Karypis, 2002] is used on these representations in order to obtain clusters of semantically related words. CLUTO is a software package for clustering low and high dimensional data sets and for analysis of the characteristics of the various clusters. CLUTO provides three different classes of clustering algorithms, based on partitional, agglomerative and graph-partitioning paradigms. It computes clustering solution based on one of the different approaches.

For this article, we are interested only in the first three steps of the DISCOveR process. Step 1 is the linguistic preprocessing of the corpus. The raw text is cleared from non-linguistic data, it is PoS tagged and syntactically parsed. In

[1]Available at: `https://code.google.com/archive/p/word2vec/`

[2]Available at: `http://wacky.sslmit.unibo.it`

Step 2, the DSM matrix is constructed. The rows of the matrix correspond to lemmas and the columns correspond to contexts. Contexts in this approach are defined as a triple of syntactic relation, direction of the relation and lemma in [direction:relation:lemma] format[3]. This matrix is used to generate vector representations for the 10,000 most frequent words in the corpus. Next, Step 3 uses CLUTO to create clusters of semantically related lemmas from the DSM matrix and the corresponding vectors. The clusters are created based on shared contexts.

Martí et al. [2019] start from a raw, unprocessed corpus and in Step 1 they clear the corpus and tag it with the linguistic data relevant to the matrix extraction. The format they use is shown in Table 2.1.

Table 2.1 Diana-Araknion Format

Token	sanitarios
Lemma	sanitario
PoS	NCMP
Short PoS	n
Sent ID	000
Token ID	0
Dep ID	2
Dep Type	suj

The original DISCOveR experiment is done with the Diana-Araknion corpus of Spanish. For the purpose of this article, we replicated the process for English, using the PukWaC corpus. For step 1 we had to make sure that our preprocessing is equivalent to the one of Diana-Araknion. The corpus PukWaC is already preprocessed and the format is similar to the one of Diana-Araknion. However, in order to make it fully compatible, we had to make several modifications of the format and linguistic decisions. Regarding the format, we removed any remaining XML tags, enumerated the sentences in the corpus, and generated "short PoS"[4]. From the linguistic side, we had to decide whether all PoS and Dependencies were relevant for the vector generation or some of them could be merged together or even discarded in order to optimize and speed up the process.

The process of generating vectors and clusters is based on analyzing the contexts where each word appears in. A word is identified by its lemma and its PoS

[3]For example, from the sentence "El barbero afeita la larga barba de Jaime", three different contexts of the noun lemma barba are generated: [<:dobj:afeitar_v] , [>:mod:largo_a] and [>:de_sp:pn_n]. The example is from Martí et al. [2019]

[4]short PoS is a one letter tag representing the generic PoS tag of the lemma. In this experiment, short PoS is the first letter of the full PoS

tag. However, in the PukWac tagset there are many PoS tags which specify not only the PoS 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 different PoS tag. To avoid this problem and to generate only one vector for all of the different word forms, we have decided to merge certain PoS tags under one category.

We decided to simplify the POS tagset further. It is a common practice in DSM to focus the experiment on the relations between content words. Function words and punctuation are usually not considered relevant contexts. Because of that, we have put them under the common tag "other". All of the changes on the PoS tagset are summarized in Table 2.2.

Table 2.2 PoS tagset modifications

Tag	Original tag	Description
J	JJ JJR JJS	Adjective
M	MD	Modal verb
N	NN NNS	Noun (common)
NP	NP NPS	Noun (personal)
R	RB RBR RBS RP	Adverb
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

The list of syntactic dependencies in PukWaC is also not fully relevant to the task of vector generation. While the unnecessary PoS tags may lead to multiple vectors for the same word, unnecessary dependencies generate additional contexts, increasing the dimensionality of the vectors and leading to a more complicated computational process. Therefore the modification of the dependencies is mostly related to the optimization of the computational process. After analyzing the tagset, we have decided to merge the **OBJ** and **IOBJ** tags due to some inconsistencies of their usage. We have also decided to discard the following relations: **CC** (conjunction), **CLF** (be/have in a complex tense), **COORD** (coordination), **DEP** (unclassified relation), **EXP** (experiencer in few very specific cases),