

Faculty of Philosophy, Arts and Letters

Designing and evaluating the Phraseology Extraction Package with a focus on nested n-grams and variation

Author: Laurane Castiaux
Supervisor: Dr Magali Paquot
Academic year: 2020-2021
Master in Linguistics with a professional focus on Natural Language
Processing

Table of Contents

Chapter 1	Introduction.....	3
1	Phraseology.....	3
2	Practical applications of phraseology.....	7
2.1	Word combinations in NLP.....	7
2.2	Word combinations in other linguistic fields.....	8
3	Automatic word combination extraction.....	9
4	Objectives and outline of the thesis.....	10
Chapter 2	State of the art.....	11
1.	Cluster analysis.....	11
1.1	Definition and methodology.....	12
1.2	Implementation.....	14
1.3	Available tools.....	22
1.4	Conclusion.....	25
2	Co-occurrence analysis.....	26
2.1	Definition and methodology.....	26
2.2	Implementation.....	28
2.3	Available tools.....	36
2.4	Conclusion.....	38
3	Conclusion.....	39
Chapter 3	Phraseology Extraction Package.....	41
1	Implementation.....	41
1.1	The ideas behind PEP.....	42
1.2	Extraction pipeline.....	51
1.3	Short examples of PEP queries and results.....	55
2	Evaluation.....	58
2.1	Preparing the data.....	61
2.2	Quantitative analysis.....	61
2.3	Qualitative analysis.....	64
2.4	Discussion of the results.....	73
3	Limitations & future work.....	74

Conclusion.....	75
Acknowledgments.....	76
References.....	77

Chapter 1

Introduction

In the last decades, the advent of corpora and corpus linguistics have shown that the language is not only constituted by words, but also by “semi-preconstructed phrases” (Sinclair 1991) or “word combinations”; this is the object of study of phraseology. Since then, linguists have not been able to reach an agreement on how to identify those word combinations, nor on how to define them. As a result, many different definitions and approaches to phraseology exist and automatic word combination extraction tools must reflect this diversity. The objective of this thesis is to describe and evaluate a new highly customizable tool to automatically extract word combinations. This tool focusses on extracting word combinations of different lengths, and on integrating more variation in the word combinations by taking into account not only tokens, but also lemmas and part of speech tags.

1 Phraseology

Phraseology is the part of linguistics that studies word combinations. Word combinations can take many forms: they can be long (e.g. “When life gives you lemons, make lemonade”), short (e.g. “thank you”), very fixed (e.g. “a stitch in time saves nine”) or more flexible (e.g. “door + knock” in “knock on the door”, “the door I knocked on”, etc.). Many sub-fields of linguistics have contributed to the study of word combinations, and yet there is no consensus on what they are or on how to identify them. This section gives an overview of the different definitions and terminology given to phraseology and word combinations as well as the various features that can be used to identify them.

Two main approaches to phraseology co-exist: the first one is referred to as the “phraseological approach” (Nesselhauf 2005) or the “traditional approach” (Granger & Paquot 2008), and it focusses on word combinations whose meaning is not transparent, and that show a high degree of syntactic and lexical fixedness. It gives a linguistic definition of word combinations and tries to categorise them. The second approach is called the “frequency-based approach” (Nesselhauf 2005) and it studies word combinations from a frequency or statistical perspective.

Mel’čuk, a main representative of the traditional approach, calls word combinations *phrasemes* and defines them as follows:

[...] the main substantive property of a phraseme is its non-compositionality: it cannot be constructed [...] from words or simpler phrases according to general rules of L [a given language], but has to be stored and used as a whole. (Mel’čuk 2001: 24)

According to this definition, any group of words whose meaning is **not compositional** – i.e. whose meaning cannot be inferred from the meaning of its constituents – is considered to be a *phraseme*. Non-compositionality is often associated with **fixedness**. A word combination is fixed if it cannot undergo linguistic transformations – such as insertion, substitution and passivation – without fundamentally changing its meaning. For example, if we replace the word “kick” by one of its general synonyms, “hit”, in the word combination “kick the bucket” (i.e. “hit the bucket”), this phrase loses its meaning of “die”. Fixedness is a continuum and not a clear-cut characteristic of word combinations; a word combination might support one transformation and not another, and it can be the opposite for another word combination. It is also not always clear what transformations a word combination can or cannot undergo. For example, “spill the beans” can sometimes undergo insertion, as in “spill the political beans”, but not always (“*spill the small beans”). Another characteristic of phrasemes often mentioned is that they are **institutionalized**. Moon (1997: 44) defines an institutionalized unit as “conventionalized in the language” and “regularly considered by a language community as being a unit”. However, intuition of native speakers plays a big part in this definition and, since intuition has been compared to the reality of the data through corpora, it is widely accepted that intuition about language does not reflect the reality (Sinclair 1991).

Several linguists have tried to categorise phrasemes and to give them a more precise linguistic definition. For example, A.P. Cowie (1981; 1994), another main representative of the traditional approach, has defined several sub-categories of phrasemes, among which we find *idioms* (semantic units that resist replacement of their components, e.g. “kick the bucket”), *collocations* (associations of several lexemes that co-occur in specific grammatical construction, e.g. “heavy rain”) and *formulae* (word combinations that have a discourse function, e.g. “good morning”). Those categories

can help study and understand how words combine together, but they are not mutually-exclusive and have hazy borders.

With the advent of corpora and corpus linguistics, it became easy to count the **frequency** of words and groups of words; word combinations could then be analysed from a frequency and statistical perspective. This is what is referred to as the frequency-based approach to phraseology. In this movement, J. Sinclair defines a *collocation* as “the occurrence of two or more words within a short space of each other in a text” (1991: 170), this corresponds, for example, to “heavy rain” or “knock + door”.

With this definition and access to corpora, it became possible to automatize the identification of collocations by counting the groups of words occurring near each other and selecting the most frequent ones. Sinclair gives a cognitive explanation for this phenomenon: the language would not only be constituted by individual words, but also by semi-preconstructed phrases that the speaker can access and retrieve as a whole. He calls this the *idiom principle* and opposes it to the *open-choice principle*, which is the fact of constructing a phrase word by word. Building a sentence would be a balance between those two principles, sometimes retrieving a phrase as a whole and sometimes picking individual words. This theory and the new definition of collocations have broadened the field of phraseology: it has highlighted the omnipresence of word combinations, even when they are semantically transparent – such as “knock + door”. Thanks to lemmatized and part-of-speech-tagged corpora, it also became possible to analyse phraseology through lemmas¹ and parts of speech instead of tokens². Sequences, such as NOUN + VERB + PREPOSITION, are useful to have an insight on the grammatical patterns that are typical of a corpus. I will come back to this in Chapter 2 section 1.2.1.

The frequency-based approach relies on two main methodologies to identify word combinations: cluster analysis and co-occurrence analysis. Cluster analysis focuses on recurrent continuous sequences of words while co-occurrence analysis uses association measures to compute how strongly two or more words are associated – those that are most associated are considered to be phraseological. These two methodologies are the main methods used in natural language processing (NLP) to identify automatically word combinations; they will be described thoroughly in the chapter State of the Art.

Many sub-fields of linguistics have contributed to the study of word combinations. An important one that has not been mentioned yet is psycholinguistics. Many authors have defended the point of view that word combinations are **processed as single units** by the mind (e.g. Biber et al. 1999; Ellis 1996). In psycholinguistics, word combinations are

1 Lemmas are the canonical form of words; they correspond to the entries of a dictionary.

2 Tokens are character strings, usually defined by a regular expression; they roughly correspond to orthographic words.

defined as groups of words that are stored in the mind as a whole, and retrieved as such; word combinations are thus retrieved faster than words that are not combined.

In the store of familiar collocations there are expressions for a wide range of familiar concepts and speech acts, and the speaker is able to retrieve these as wholes or as automatic chains from the long-term memory; by doing this he minimizes the amount of clause-internal encoding work to be done and frees himself to attend to other tasks in talk-exchange, including the planning of larger units of discourse. (Pawley & Syder 1983:192)

It has indeed been proven that words that have some kind of association with the surrounding ones are processed faster than the others (Tremblay et al. 2011; Meyer et al. 1975; Christiansen & Arnon 2017; Saito 2020).

The study of **code-switching** also offers a possible way to identify word combinations. Backus (2003) suggests that most code-switches are single lexical units, even though they might be constituted by several morphemes. He cites examples of Dutch word combinations inserted into Turkish speech, such as compound nouns (e.g. “cremespoeling”, literally *cream rinse*, i.e. *hair conditioner*), collocations (e.g. “verkorte opleiding”, *short program*) and discourse markers (“vind ik”, *I think*). He insists that not all contiguous code switches constitute a single unit, but argues that the vast majority of them do. As a result, code switches may be used as a hint to identify word combinations, but is not a sufficient condition.

The last criteria that I will mention come from **phonology**. Peters (1983) is the first to use what she calls *phonological coherence* as a criterion, among others, to identify word combinations. She defines a phonologically coherent sequence as fluent, that is with an unbroken intonation contour, and free of hesitations. However, in an exploratory study, Lin & Adolphs (2009) have shown that phonological coherence can help identify word combinations in only 55% of the cases. It should thus never be used alone as only identification criteria. Wray (2002) mentions two other phonological criteria to identify word combinations: the stress and the liaison in French. About the stress, she cites Jespersen (1924/1976) who argues that, since word combinations are considered to be a single unit, they cannot be stressed in an unaccustomed way. As for the liaison, in French, it is seen as a sign of correctness and is mostly used in formal speech. Bybee (1998) suggests that, when it is used in colloquial speech, it might be the sign of a unit stored and retrieved as a whole, with no extra processing effort to produce the liaison.

All the criteria mentioned to identify word combinations can be used together to identify groups of words that fit better to a given research question, or simply to correspond to the data that is analysed – e.g. code switching can only be useful if the data analysed contains code switches. Depending on the definition given to word combinations and on how they are identified, they constitute between c. 30% (Biber et

al. 1999) and c. 80% (Altenberg 1998) of the English language. The difficulty to pinpoint precisely what word combinations are due to the fact that there is no clear-cut distinction between “random” associations of words and phraseological word combinations; phraseology is a continuum and there is no dichotomy in language that would distinguish what is phraseological from what is not.

2 Practical applications of phraseology

Word combinations are not only interesting as the object of study of phraseology, they also have many applications in various fields of linguistics. This section gives an overview of a few of those applications, first in NLP, then in other linguistic domains.

2.1 Word combinations in NLP

As explained in the previous section, some word combinations are non-compositional, which means that their meaning cannot be inferred from the meaning of their constituents. As a result, they must be treated as a whole in any NLP task that manipulates meaning. A prototypical example of that is automatic **machine translation** (MT). If phraseology is not taken into account in a translation, it results in word-to-word translation of word combinations such as Fr. “Il a frappé le seau” for En. “He kicked the bucket”. That kind of automatic literal translation was still common until recently. According to Monti et al. (2020), the best way to handle word combinations in MT is to translate texts in which the word combinations have been annotated (automatically or manually). Then, those words can be treated as a single unit in the translation process. However, the most complicated task might not be to translate word combinations, but to identify them. While the phrase “kick the bucket” most often means “to die”, in some contexts it is possible that it actually refers to someone kicking a bucket. In that case, it should be translated word by word and not as a single unit.

Word combinations are also useful in **word sense disambiguation** (WSD). WSD is an NLP task that aims at attributing the right meaning to words in a text from a predefined list of meanings, typically the meanings found in the entry of those words in the dictionary. Since word combinations do not always have the same meaning as single words, it is important to identify word combinations to attribute them their real meaning. Furthermore, Finlayson & Kulkarni (2011) point out that word combinations found in a sense inventory are on average less polysemous than single words. For example, “world record” has only one meaning listed in Wordnet, while “world” has nine and “record” fourteen. By identifying what groups of words share a single meaning, the authors could improve their results in WSD.

The last example of the use of word combinations in NLP is for **language generation**. As mentioned in the previous section, word combinations are everywhere in language. This means that a program that generates text or speech in a natural language cannot ignore them. The main problem related to word combinations in language generation is to choose the right word combination. As explained by Smadja & McKeown (1991:230): When requested to produce a sentence expressing the fact that the tea in question has been brewed for a long time, a language generator has no a priori reason to favour the adjective strong over powerful. It must have access to interlexical constraints on these words to use them correctly.” As a consequence, the language generator cannot only rely on the semantic of individual words to select the right one. Instead, it must have access to typical word associations to restrict its possible candidates to what would occur in a speech produced by a native speaker.

2.2 Word combinations in other linguistic fields

Outside of NLP, word combinations have been studied from several perspectives.

For example, they have been studied by lexicographers to include them in **dictionaries** and describe their semantic and syntactic properties. Word combinations represent two challenges for lexicographers: their identification and their description. As presented, word combinations is a generic term that groups a wide variety of linguistic phenomena. Lexicographers must identify the word combinations that would be useful for their purpose; the word combinations found in a domain-specific dictionary will not be the same as those explained in a learner dictionary. It has been shown that word combinations are particularly hard to master for speakers of a foreign language, they are thus particularly important in learner dictionaries (Granger & Paquot 2010). Once identified, the word combinations need to be explained and described, once again, in a way that is relevant to the lexicographer’s objectives. In the case of a learner dictionary, the usage of the word combination is particularly important, while it may be obvious to native speakers. Those descriptions may include information about the meaning, the context of usage and the syntactic properties of the word combinations.

Word combinations play a major role in second **language acquisition**: they are difficult to acquire for non-native speakers and yet they are essential to mastering a language. Extensive research has been conducted to understand better learners’ difficulties related to word combinations, and teachers have thought of new ways to teach vocabulary to include word combinations, such as Lewis’ lexical approach (Lewis 2000) or corpus-based methods (Ackerley 2017). Like lists of vocabulary words were made, that should be learned by students of a language (e.g. the General Service List (West 1953)), researchers built similar lists with word combination, such as the Academic Collocation List (Ackermann & Chen 2013) that focuses on collocations as defined by (Sinclair

1991) and the Academic Formulas List (Simpson-Vlach & Ellis 2010) that contains word combinations specific to academic English.

3 Automatic word combination extraction

Because of the many different definitions given to word combinations, most of them being quite vague, it is a challenge for linguists to identify them. Computational linguists have tried for the last decades to make this identification easier, faster and more systematic by relying on corpora and computational techniques. Because the process of automatic retrieval of word combinations is completely different from what a non-computational linguist would do to identify them manually, I choose to use the term “multi-word units” (MWUs) to refer to automatically extracted word combinations, and the term “phraseological expressions” to refer to linguistically defined word combinations. Ideally, MWUs should reflect the same reality as phraseological expressions but, in practice, this is not the case because of the different criteria used for their identification: whereas linguists use criteria such as non-compositionality, resistance to linguistic transformation or intuition, computational techniques rely mostly on frequency of occurrence and co-occurrence of groups of words. Furthermore, as suggested by Granger & Paquot (2008), the extraction and the study of word combinations are two different tasks and linguistic categories should not be used to refer to automatically extracted word combinations; their classification should remain the responsibility of the linguist and not of a computer program.

A wide spectrum of methods and tools can be used to extract MWUs and they differ greatly from each other. Some rely exclusively on information contained in a raw text (mainly frequency of (co-)occurrences of words), others rely on taggers or parsers to add linguistic information to the texts and take it into account for the extraction, and finally some use linguistic transformations and the fixedness criterion to identify them. The main advantage of using raw texts is that it does not require any language specific information and thus offers a “ready to use” method for any language. But mathematical models are not perfect, and in the case of MWU extraction, they tend to underestimate the importance of low frequency words. Furthermore, most statistical extraction methods rely on arbitrary thresholds. Ideally, every choice should be linguistically motivated and not be made purely for a practical reason. Using linguistic information has shown to greatly improve the accuracy of MWUs extraction. Part of speech tagging can disambiguate between words, lemmatisation can group together different realization of a same MWU, and parsing can ensure that the words selected as MWUs are syntactically related, even if they are separated by many words.

The problem of the tools available to non-computational linguists to automatically extract MWUs is that they do not reflect the diversity of definitions of phraseological

expressions. As we will see in the next chapter, most tools only offer one way to extract MWUs, with very few customizable parameters, which restricts the kind of analysis that non-computational linguists are able to conduct on corpora. I will attempt to answer those limitations by programming a new highly customizable tool for automatic MWU extraction, with a focus on internal variation of the MWUs by integrating the use of lemmas and part of speech tags inside the units, and on nested MWUs, i.e. MWUs that are comprised inside others (e.g. “at the same” which is nested inside “at the same time” and “at the same place”).

4 Objectives and outline of the thesis

This thesis aims at describing and evaluating the Phraseology Extraction Package (PEP), a command-line tool that I designed for automatic MWU extraction, mostly during my internship at the Center for English Corpus Linguistics at the Université Catholique de Louvain. This tool has the main objectives to offer a highly customizable extraction process, to offer a relevant way to display nested n-grams, and to integrate not only tokens, but also lemmas and part of speech tags into the MWUs. Chapter 2 is a review of the two most used techniques to automatically extract MWUs, namely the cluster analysis (section 1) and the co-occurrence analysis (section 2) and, for each technique, I will present tools that are currently available to non-computational linguists and their limitations. Chapter 3 will first describe how PEP works and how it was implemented (section 1), then evaluate it by running it on several corpora and analysing the results (section 2), and finally present the current limitations of PEP and potential future improvements (section 3).

Chapter 2

State of the art

This state of the art presents two different methods to automatically extract MWUs that come from corpus linguistic studies: the cluster analysis and the co-occurrence analysis. In those two methods, a list of MWU candidates is made from a corpus, which is then filtered to keep the final list of MWUs, which are supposed to be phraseological expressions. The focus is thus on the extraction process and not on the analysis of the MWUs. There exist other extraction methods based on machine learning (e.g. Yang 2003); those methods are not the subject of this thesis.

Each section of this state of the art starts with a definition and explanation of a general methodology to extract MWUs. It is followed by a discussion of more technical aspects to take into account to implement the automatic extraction and their influence on the extracted MWUs. Finally, a short review of available tools to extract MWUs is made.

1. Cluster analysis

The first method presented in this chapter is the cluster analysis. I will start by defining the MWUs studied with that methodology and then explain how such sequences can be automatically extracted. To do so, I will detail all the parameters that can be taken into account for the extraction as well as their impact on the results. Finally, I will present the tools that are available to non-computational linguists to extract MWUs with this methodology.

1.1 Definition and methodology

Cluster analysis studies phraseological expressions in the form of recurrent sequences of linguistic elements, usually tokens, lemmas or part of speech (POS) tags. Some possible examples of such sequences are *as a matter of fact*, *I think that* and VERB PREPOSITION DETERMINER NOUN. In this methodology, the main criterion to define a sequence as phraseological is its frequency: all sequences occurring more than a given number of times can be MWUs. Once the candidates are selected, non-relevant sequences can be filtered out, for example according to their dispersion through the corpus or according to some linguistic filters, such as the POS of the words constituting the MWU.

Those MWUs have received many names from different authors such as *recurrent word combinations* (Altenberg 1998), *lexical bundles* (Biber et al. 1999), *chains* (Stubbs & Barth 2003), *recurrent sequences* (De Cock 2003) or *clusters* (Scott & Tribble 2006). The extraction of such sequences can be summarized as follows:

Step 1 Segment the corpus into linguistic elements

Step 2 Count all sequences of n elements (= n-grams = MWU candidates)

Step 3 Filter out non-relevant sequences

In Step 1, the corpus is pre-processed, usually by tokenizing it and sometimes by applying a POS tagger on it. The linguistic elements that are used in this methodology are usually tokens or POS tags, since those are the elements that constitute phraseological expressions. However, it may be interesting to focus on morphemes or letters for some applications; this will be detailed in section 1.2.1. Then, in Step 2, all consecutive sequences of n elements, called n-grams, are extracted and counted. The n-grams are our MWU candidates. It is possible to choose in turn several values of n to extract sequences of different sizes. Finally, in Step 3, the candidates are filtered to keep only those that we want to be considered as MWUs. The main filter in this methodology is to set a frequency threshold: if a candidate appears frequently enough, it can be considered as a MWU. The idea behind it is that more frequent sequences are more salient to the mind and are processed as a single unit, faster than compositional phrases (Ellis 2002; Arnon & Snider 2010). Furthermore, Leech (2011) has demonstrated that frequency has a strong influence in language learning: words that are used more frequently are learnt faster. When using frequency alone as a filter, incomplete sequences (e.g. *on the other*) or sequences made of only grammatical words (e.g. *is in the*) might be identified as MWUs. To prevent this, it is possible to use lists of stop-words, i.e. words that should not appear in the MWUs, or to limit the results to some linguistic patterns, such as VERB + PREPOSITION + DETERMINER + NOUN. Another common filter is to select only sequences that are dispersed homogeneously through the corpus. Filters will be detailed in section 1.2.3.

These steps are a generalisation of the most commonly used methodology in the cluster analysis, and the results vary according to how these steps are operationalised. The different possible implementations and their impact on the results are explained in the section 1.2.

A limitation of the strict definition given above is the lack of internal variability allowed by the sequences. Some sequences, such as *a * of*, are very frequent, but each one of their realisations, e.g. *a group of* or *a quarter of*, might not be frequent. Renouf & Sinclair (1991) understood that and studied what they called *collocational frameworks* which they defined as “discontinuous sequence[s] of two words, positioned at one word remove from each other” (1991: 128). This notion of sequences with one variable slot was taken up later by Fletcher (2003-2011) who called them *phrase-frames* or *p-frames*. To extract p-frames, after counting each n-gram, they are compared to each other and combined if they differ in only one element.

The first authors to use token n-grams to identify MWUs are (Choueka et al. 1983). They were interested in any “sequence of two or more *consecutive* [emphasis in original text] words that constitutes an autonomous linguistic unit and has acquired, because of recurrent use in specialized contexts, a meaning or a connotation that somehow transcend the ordinary meaning of its constituents” (1983: 34). To identify such sequences, they developed an algorithm which constructs n-grams from two to six words, selects those occurring at least ten times, and filters them according to several lists of stop-words to ensure that the retrieved sequences would be complete grammatical units. Choueka (1988) tested the algorithm on the New York Times News Service, a ten million word corpus and obtained very relevant results, such as “catholic church”, “federal government”, “changed his mind”, “played a role” and “value of the dollar”.

Many years later, Biber and his colleagues made researches on what they call “lexical bundles”. They wrote their results in a chapter of the *Longman grammar of spoken and written English* (1999) and it had a great influence on the study of MWUs. They define lexical bundles as “recurrent expressions, regardless of their idiomaticity, and regardless of their structural status.” (1999: 990). To list them, they selected n-grams with three or more words from the LSWE corpus, and put a frequency threshold of at least ten occurrences per million words (three- and four-grams) or five occurrences per million words (five- and six-grams). Moreover, they added a dispersion criterion: for a sequence to be a lexical bundle, it has to appear in at least five different texts in a register. They retrieved sequences such as “the end of the”, “I don’t know what it”, “That’s what I mean”, “go and have a look”, “going to do it”, “what did you do”, “all over the place”. This was followed by a detailed analysis of the lexical bundles across different genres.

1.2 Implementation

The steps presented above can be operationalised in different ways to adjust the resulting MWUs to correspond to one of the many possible definitions of phraseological expressions. This section will present different possible implementations of each step of the general methodology as well as the influence of each choice of implementation on the results.

1.2.1. *Choosing the elements constituting the MWUs*

In the cluster analysis, the corpus preprocessing consists of the segmentation of the corpus into linguistic elements and sometimes of tagging it with linguistic information. We must thus consider what linguistic elements are the most suitable for our purpose: characters, tokens, lemmas, tags, or a mix of different elements.

Identifying recurrent sequences of **characters** has proved to be useful in some NLP applications, such as optical character recognition (Chang & Chen 1995; Palkovic 2008) or language identification (Kulmizev et al. 2017). This level of analysis is however too small to be useful for a phraseological analysis.

In case the focus lies on the meaning of the MWUs, words (**tokens or lemmas**) are probably the most suitable units since they bare meaning. The advantage of using lemmas over tokens is that it would then group, for example, the sequences *I do not know* and *I did not know* under the same MWU *I do not know*. It might be useful to keep such sequences separate to analyse the style of a corpus more closely and not miss, for example, specificities in the use of time. In the literature, token n-grams are much more used than lemma n-grams (e.g. the very influential Biber et al. (1999)).

If we want to focus on grammatical phenomenon, **POS tags** are probably the most appropriate. They were used, for example, by Aarts & Granger (1998) to identify the footprint of learners of English as a foreign language. They looked at the most frequent POS three-grams in the Dutch, Finnish and French parts of the Corpus of Learner English and in the Louvain Corpus of Native English Essays, which was taken as reference. Those corpora were tagged with the TOSCA-ICE tag-set, which contains 270 different tags. They found that the top twenty three-grams are the same in all the four sub-corpora, but with different ranks: all the learner groups tend to overuse patterns starting with a connective and to underuse those involving prepositions. Some patterns are more specific to one learner population, for example the French-speaking learners overuse sentence-initial to-infinitive clauses of purpose. Another example is the study of Cappelle & Grabar (2016) who wanted to create an “n-grammar” of English for teachers of English as a foreign language, for their students and for material developers. They believe that “(...) grammatical n-grams are constructions, in a Construction Grammar sense: they are form-function pairings which native speakers have memorized (and

which learners of a language should acquire) as a result of their high frequency” (2016: 273-4). Therefore, they wanted to create a list of such constructions that could help to learn English as a foreign language. To do so, they selected the 100 most frequent POS five-grams from the Corpus of Contemporary American English. When it was necessary, they added an element to the right of the five-gram to ensure that the result was a complete grammatical phrase (e.g. they added a noun after the sequence *the adj X_{noun} of the*). The tag set used was the CLAWS 7, constituted from 137 word categories. They argued that it was the right level of precision for their purpose. The resulting list was accompanied by examples to illustrate how the constructions can be used. The number of different tags used on a corpus will have a great impact on the number of different POS n-grams that will be found. By adjusting the precision of the tag set, we can focus either on more specific phenomena or, on the contrary, have a general overview of POS patterns present in a corpus. The number of different results obtained will vary accordingly. The inconvenience of POS tags is that they require more language-specific text pre-processing and, when done automatically, it may not always be completely accurate.

Very few authors have studied MWUs made of **both words and tags** although they can be very relevant from a linguistic point of view, since tags can be used to fill in a free slot in a word sequence. This is a way to give more information about the MWUs while allowing for some variation. It would be possible to automatise this process by extracting p-frames and, if no lexical variant but a tag prevails, use it to fill the slot. Both POS or semantic tags could be used for that purpose.

1.2.2. *Choosing the size of the sequences*

After choosing what elements will be part of the MWU candidates, we must decide how many elements will constitute them. The most relevant size to study phraseological expressions is between three and six words. The preferred length is four-word sequences because they are common, they represent a large range of structures and are few enough to be manually analysable (Hyland 2008; Chen & Baker 2010). MWUs of 6 or more words can turn out to be statistically non-significant, that is MWUs that do not correspond to phraseological expressions (O'Donnell et al. 2013).

By restricting the results to only one size, many potentially interesting MWUs are omitted. But selecting MWUs of several sizes at the same time will include nested sequences, i.e. shorter sequences (e.g. *on the other*) that are part of longer ones (e.g. *on the other hand*). There are two ways to improve the resulting list: either the sequences have to be sorted in an order that is easy to read, by grouping related sequences, or the less relevant units have to be filtered out. For the first possibility, I will present the solution of Lefer & Grabar (2015) and Dias (2002) and, for the second, the *entropy value* developed by Shimohata et al. (1997) and the filtering method of (Merkel & Andersson 2000).

Lefer & Grabar (2015) have suggested to group various size MWUs into “bundle families”. They defined a bundle family as a group of various size n-grams that all contain the same shorter n-gram. Dias (2002) introduced the same notion as “sur-groupe d’un N-gramme” (n-gram super-group) and “sous-groupe d’un N-gramme” (n-gram sub-group). The super-group of an n-gram is all the longer n-grams containing that n-gram. The sub-group of an n-gram is all the shorter n-grams found in that n-gram. I call super-n-grams all the n-grams of a super-group and sub-n-grams all the n-grams of a sub-group. There are several advantages of grouping various size n-grams in such a way. First, it gives a glimpse of the use of the shorter units by showing what words often precede or follow them. Next, it gives an easy-to-read overview of the extracted sequences since they are listed in an easily readable order (unlike pure alphabetical order, frequency, or n-gram size). It therefore eases the manual vetting, and the comparison of various size MWUs across different (sub-)corpora, which means that it can help in cross-genre or cross-linguistic analyses. Unfortunately, today, no available tool offers to sort various size MWUs in a similar way.

It is also possible to compare the variants of a bundle family, and only keep the most relevant sequences. To do so, (Shimohata et al. 1997) consider that MWUs occur in a noisy environment, as defined in information theory (Shannon 1948). He computes the entropy value of each side of a sequence according to Shannon’s formula for entropy (Equation 2.1) and if the entropy of at least one side is below a given threshold, the sequence is discarded.

$$H(str) = \sum_{i=1}^n -p(w_i) \log p(w_i) \quad (\text{Eq.2.1})$$

where *str* is a string identified as a MWU; $w_1 \dots w_n$ are all the possible adjacent words of *str*; $p(w_i) = \frac{\text{freq}(w_i)}{\text{freq}(str)}$ is the probability of each possible adjacent word.

As highlighted by Silva et al. (1999), this formula favours sequences that begin or end with very frequent words such as determiners or pronouns. If it is applied on a whole corpus to identify MWUs, as the authors intended it to be, some units such as compound nouns would probably be missed. To avoid this, the formula can be applied only on previously identified sequences that are nested in longer ones.

Merkel & Andersson (2000) used a much simpler algorithm to filter various size MWUs: they removed all the n-grams that had the same frequency as an n-gram of their super-group. This method keeps exactly the same MWUs as the entropy value with a threshold of 0. With a higher threshold, the entropy value would reject more MWUs.

The advantage of sorting nested sequences is that all the MWUs found are given to the linguist for analysis. The advantage of the entropy value is that it gives only the MWUs

that are considered the most relevant, greatly reducing the need for a manual vetting. The solution found in Merkel & Andersson (2000) is a compromise between the two other methods.

1.2.3. *Filters*

There are three kinds of filters that can be applied to the MWU candidates of the cluster analysis: filters based on frequency, on dispersion or on linguistic characteristics. There are also two filters that are specific to p-frames. Each of those filters will be discussed in turn in this section.

Frequency is by far the most used criterion to filter out non-relevant multi-word sequences and yet there is no consensus on how it should be used. There is psycholinguistic evidence that more frequent sequences are processed faster, both by native speakers (Arnon & Snider 2010; Tremblay et al. 2011) and by learners (Leech 2011). This is a good argument to use frequency as a criterion to identify MWUs. Frequency has a continuous effect: each occurrence of a word or group of words makes it more entrenched in the brain and there is no evidence for a threshold beyond which recurrent sequences are processed faster. However, to use the frequency of candidates as a filter, a threshold must be defined. Even though frequency thresholds do not reflect the language processing of the brain, research such as the influential one by Biber et al. (1999) has shown that it is very easy to implement and works quite well to identify sequences that are interesting to analyse from a phraseological perspective. The difficulty of setting the right threshold lies in finding a balance between setting a threshold high enough to avoid selecting sequences that occur often only by chance, and low enough to identify all the sequences that could be relevant for the study.

The most straightforward method is to set a raw frequency threshold. This is the approach used in most studies, but it can be used only to work on one corpus or on corpora that all have the same size; a sequence of words is more likely to be more frequent in a longer corpus than in a shorter one.

To overcome this limitation, researchers often select a relative frequency threshold, usually expressed as a number of occurrences per million words. As underlined by Chen & Baker (2010), Bestgen (2018) and Pan et al. (2020), it is important to keep in mind that, even if we manipulate relative frequencies, the results obtained with a given threshold will be influenced by the size of the corpus. Hence, MWUs extracted from (sub-)corpora of different sizes with the same threshold should be compared with caution. Another solution is to compare a given number of the most frequent sequences of each (sub-)corpus, for example the 100 most frequent sequences, as did Fletcher (2004) or Garner (2016).

Of course, the frequency of a sequence is inversely related to the number of words it contains; a sequence of 4 words can at most be as frequent as the sequences of 3 words it contains, it can never be more frequent. As a result, if MWUs of different lengths are analysed at the same time, a different threshold must be set for each sequence size, with lower thresholds for longer sequences. Furthermore, in such a case, O'donnell (2011) suggests the use of *adjusted frequencies* instead of raw or relative frequencies. The adjusted frequency of a sequence corresponds to its raw frequency minus the number of occurrences of the longer sequences in which it appears. For example, if “you know” appears 10 times in a corpus and “know” appears 25 times, the adjusted frequency of “know” would be $25 - 10 = 15$.

As underlined by Bestgen (2018), a sequence does not need to be statistically significant to be interesting from a linguistic point a view: some sequences, even very infrequent ones, can be highly idiomatic and thus be relevant for some studies. By putting a threshold as low as 2 occurrences, we ensure to retrieve all sequences that could be interesting, but this increases the amount of noise in the results.

Frequency alone is not enough to tell how common words or expressions are in a language. For example, the word combination “rheumatoid arthritis” may be very frequent in a medical paper about arthritis, but will most probably be absent in any other context. The notion of **dispersion** of a word or word combination refers to this phenomenon. If a word is homogeneously distributed in a corpus, it is representative of this corpus; on the contrary, if it appears in only one of its parts, it is specific to that part, and not representative of the whole corpus, even if it is very frequent. Dispersion measures have been developed to quantify this phenomenon, and that is why dispersion is usually used next to frequency to identify and sort MWUs.

Dispersion measures have been developed to compute the dispersion of single words in a corpus, but they work exactly in the same way with MWUs instead of words. Furthermore, we know from psycholinguistics that the ease of a word processing is not only linked with the frequency of occurrence, but also with the variety of contexts in which the word appears (Verkoeijen et al. 2004; Ambridge et al. 2006; Adelman et al. 2006). Gries (2008) has demonstrated a correlation between, lexical decision times and the dispersion of a word in a corpus. Dispersion measures are thus not just a “trick” to retrieve automatically MWUs, but they reflect some psycholinguistic reality.

The simplest way to take the dispersion into account to identify MWUs is to use the range, i.e. the number of corpus sections in which it a sequence occurs. To identify MWUs that belong to general expressions of English (and not to a particular domain), many researchers (e.g. Biber et al. 1999) have set a minimal range value that sequences must reach to be selected as MWUs.

The range works well as a filter, but it does not give an accurate measure of the dispersion of a sequence since it does not take into account the number of occurrences of the MWU in each sub-corpus. Therefore, more complex dispersion measures were developed. A list of such measures can be found in Gries (2020). I will present here the two most used dispersion measures: Juilland’s D and Gries’ DP .

Juilland’s D (Juilland & Chang-Rodriguez 1964) is based on the coefficient of variation of the frequency of a word in each corpus section. It is defined as follows:

$$D = 1 - \frac{CV}{\sqrt{n-1}} \text{ where } CV = \frac{\sqrt{\frac{\sum (x_i - \bar{X})^2}{n}}}{\bar{X}} \quad (\text{Eq. 2.2})$$

where CV is the coefficient of variation of the frequency in each corpus section (x_i) around the mean frequency in all sections ($\bar{X}$), and n is the number of different corpus sections. A D of 1 represents the most homogenous dispersion, 0 the most uneven one. Although it has been widely used, this measure has some problems: (i) it can break its theoretical minimal value of 0, (ii) it does not take into account the relative size of the sub-corpora, (iii) it over-evaluates the homogeneity when many corpus sections are used, and (iv) different distributions can result in the same value of D .

A variant of Juilland’s D (Gries 2008) has been proposed to overcome problem (ii): compute CV with the proportion of the sub-corpus containing the word, instead of the raw frequency. However, the other problem remains unsolved.

DP (Gries 2008) answers those problems and it is the most promising association measure today (see Biber et al. (2016) for a comparison of D and DP). It is based on the difference between the expected proportions ($Pexp_i$) and the observed proportions ($Pobs_i$) of the word in each corpus section (Equation 2.3). When DP equals 0, it means that the word is perfectly distributed through the different corpus parts; when it is close to 1, the word has an uneven dispersion.

$$DP = \frac{\sum_{i=1}^n |Pexp_i - Pobs_i|}{2} \quad (\text{Eq. 2.3})$$

As explained by Egbert et al. (2020), dispersion should not be computed from arbitrarily divided corpus parts. Each section should represent a linguistically meaningful unit, such as texts, registers or academic disciplines. They argue that dispersion results should always be reported as “dispersion across” followed by the chosen type of corpus unit.

Information about dispersion can be used in different ways: it can be shown next to the extracted MWUs, it can be used to filter them in the form of a threshold, or dispersion

measures can be used to weigh the frequencies. In the last case, the frequency is generally multiplied by the dispersion.

In the cluster analysis, next to frequency and dispersion, **linguistic filters** are used to ensure that the MWUs are complete grammatical sequences. There are two ways to do so: use stop-lists or a parsed corpus. The former method is the most used because it is easier to implement.

Stop-lists are lists of linguistic elements (words, punctuation marks, POS tags, etc.) that should not be included in the MWUs. In the cluster analysis, when they are used, they almost always include punctuation marks. When the MWUs consist of only two elements, function words are often added to avoid to identify very recurrent, but yet meaningless, sequences such as “in the” or “and a”. Merkel & Andersson (2000) have made different stop-lists for words that should not appear at the beginning, in the middle, or at the end of a sequence. Making several lists has proved to be a simple and efficient method to extract complete grammatical units. Stubbs & Barth (2003) have included in their stop-list the most common English words. This filtered out MWUs that would be widely used in English, and restricted the results to sequences that were more specific to the corpora used for the extraction. In that way, they could compare more easily the MWUs extracted from different sub-corpora.

The way to ensure that the MWUs are complete grammatical sequences is to use a **parsed corpus** for their extraction. A parsed corpus is a corpus that has been annotated with the syntactical structure of its sentences. If, in theory, it is the best way to ensure that all the words constituting a sequence are syntactically related, in practice, most corpora are not parsed and parsers do not work well on every (variety of) language. Since the stop-list method is very easy to implement and works well for that purpose, parsers are rarely used in the cluster analysis.

Finally, there exist two filters that are specific to p-frames: the **variant-p-frame ratio** (VPR) and the **predictability measure**. The VPR has been introduced by Römer (2010). It is the ratio of the number of different lexical items occurring in the free slot of the p-frame on the total number of occurrences of the p-frame (Equation 2.4). If the resulting number is close to 0, the frame has few different variants; as it gets closer to 1, it has increasingly more variants. The VPR indicates the diversity of a p-frame. The higher the score, the more variants the p-frame has, relatively to its number of occurrences. For example, in Table 1, *in the * of* has 53 occurrences and 10 variants, which makes a VPR of 0.19, whereas *said the old ** has a VPR of 0.06 because it has only 3 variants with almost the same frequency.

The predictability measure was coined by Biber (2009) and later improved by Biber & Gray (2016). It is the number of occurrences of the most common variant of a p-frame

on the number of occurrences of that p-frame (equation 2.2). The predictability of the p-frame is proportional to the predictability score.

$$VPR = \frac{\text{number of different lexical items occurring in the free slot}}{\text{number of occurrences of the p-frame}} \quad (\text{Eq.2.4})$$

$$Predictability = \frac{\text{number of occurrences of the most common variant}}{\text{number of occurrences of the p-frame}} \quad (\text{Eq.2.5})$$

The predictability measure indicates how likely it is that the p-frame is used with its most common variant. In Table 1, *on the * of* (predictability: 0.44) is more predictable than *for the * of* (predictability: 0.31), even if they have the same number of occurrences and variants, because the most frequent variant of the former occurs more often than that of the latter.

Those two measures can be used to select the p-frames that fit best a research goal. P-frames with many variants – and, as a result, a high VPR – reveal general structures in the language, in which a lot of words can fit. When it has less variants, the p-frame is more context-specific. A threshold can be set to select only p-frames that are more or less variable. As for the predictability, a p-frame that is highly predictable is probably not very relevant from a phraseological perspective, and should be replaced by its most common variant. For example, “on the one *” is less relevant than “on the one hand” because, even if this p-frame has other variants, they are all much less probable than “on the one hand”.

A concrete example of how to use the VPR and the predictability measure is to be found in Garner (2016) who compares the English used by German learners of different proficiency levels. He hypothesized that learners with lower levels of English would rely on more fixed and predictable p-frames. To test this hypothesis (among others made in the paper), he extracted four-word p-frames and manually removed those not constituting a meaningful unit or those being a three-gram – i.e. a four-word p-frame, with a free slot at the beginning or the end of the frame. He also checked for orthographic variants (due to spelling mistakes of learners), and added their frequency to the prototypical sequence. He then used the VPR and the predictability measure to compute the degree of fixedness and of predictability of each p-frame. His results showed that the variability of the p-frames increases with the proficiency of the learners; advanced learners use more variants of p-frames than beginners. He found that the predictability of the p-frames decreases as the variability grows.

Table 1: Sample of the *p*-frames extracted with KfNgram on 4-grams from *Oliver Twist* by Charles Dickens

in the * of	53	10	on the * of	16	4
in the midst of	11		on the part of	7	
in the course of		7	on the top of	3	
in the hope of	6		on the subject of		3
in the direction of	6		on the point of	3	
in the way of	5				
in the habit of	5		for the * of	16	4
in the name of	4		for the sake of	5	
in the centre of		3	for the purpose of		4
in the middle of		3	for the benefit of		4
in the event of	3		for the love of	3	
said the old *	48	3			
said the old gentleman			24		
said the old lady		21			
said the old man		3			

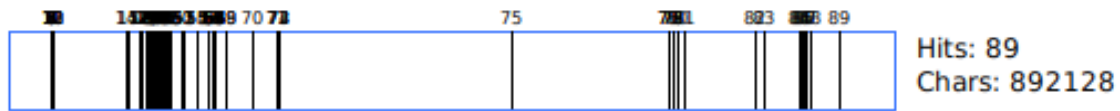
1.3 Available tools

This section gives a brief review of available tools that can be used by non-computational linguists to extract MWUs with the method described above.

1.3.1. *AntConc*

AntConc (Anthony 2019) is a multi-platform freeware that offers several tools to analyse corpora. Among those, it implements an n-gram extractor (*Cluster/N-Grams Tool*). It can extract various size n-grams at the same time and order them alphabetically or according to frequency or range (the number of files in which the n-gram is found). Clicking on a result will show the corresponding concordance lines. It can work on lemmas instead of word forms if a list of lemmas with all their inflected forms is provided by the user. The software can handle character n-grams in the same way. It is also possible to search for all sequences of a given size that contain a given sequence or that matches a regular expression. The results can be sorted in the same way as mentioned above. For each sequence given in the results, a dispersion plot can be displayed. It shows where, in each file, the sequence is used (see Figure 1 for an example).

Figure 1: AntConc’s dispersion plot for the sequence "the old gentleman" in *Oliver Twist* by Charles Dickens



Since AntConc offers to retrieve various size n-grams, it is unfortunate that it does not offer any way to handle nested sequences. It is good that it takes into account the dispersion of the MWUs by mentioning the range, however, it would be better to also implement dispersion measures. Moreover, AntConc does not support relative frequencies; all the frequencies reported by the software are raw frequencies.

1.3.2. WordSmith Tools

WordSmith Tools (Scott 2020) offers the same kind of tools as AntConc, but it is much more customizable.

One of the provided tools, *Chargram*, makes it possible to extract character n-grams of various size. It is possible to include or exclude punctuation and digits. The position of the n-gram inside the words (beginning, middle, end) is displayed along with the resulting character n-grams, and it is possible to filter them on that criterion. A list of the words in which they appear, and with what frequency, is also provided.

The tool *WordList* and its option *cluster* can make a list of n-grams of several sizes and sort them alphabetically or by frequency. The results display a dispersion value (Juilland’s D) and the range. A dispersion plot is also available. It is possible to include p-frames. The n-grams and p-frames will be mixed together in the results to follow the sorting order chosen by the user; the p-frames variants are thus not grouped together, which makes it hard to analyse and manipulate them.

WordSmith Tools is less intuitive to use than AntConc, but a good documentation is available. It offers many ways to filter the sequences and provide a lot of information about them. However, the p-frame option is hard to exploit since all the variants and p-frames are mixed in the results. It is also important to note that the chosen dispersion measure – Juilland’s D – does not take into account the relative size of the different files.

1.3.3. KfNgram

KfNgram (Fletcher 2007) allows to extract character n-grams, word n-grams and p-frames from a text file. There is no limit in the length of the sequences. If several sizes are selected at the same time, one output file will be produced by n-gram size. Every character or word n-gram is accompanied by its frequency. P-frames are accompanied

by their frequency, the number of variants, and a list of them with their respective frequency. The extracted sequences can be sorted by alphabetical or frequency order. It is up to the user to keep all punctuation marks, only the ones inside a sequence, or none. Additionally, a stop-list can be used, as well as a frequency threshold. The software can handle several files at the same time, but does not include any measure of dispersion.

KfNgram is very simple to use in its extraction of n-grams. Its main advantage is its p-frame tool, which sort the variants under their respective p-frame, and give information about the number of variants and their frequency.

1.3.4. *Sketch Engine*

The Sketch Engine is an online corpus analysis tool³. Among many other features, it offers to extract n-grams of several lengths between 2 and 6 words. Several lengths can be selected at the same time; if the user chooses to, shorter n-grams will appear right under their super-n-gram, along with their frequency, as shown in Figure 2. This appears like the opposite idea to Lefer & Grabar's (2015) concept of bundle families: bundle families group together longer n-grams that contain the same shorter n-gram whereas the Sketch Engine writes sub-n-grams under their common super-n-gram. Some filters can be set to customize the results: include or not non-words – i.e. tokens that do not start with a letter, ignore capitalization, set a minimum and/or maximum frequency threshold, matching a regular expression, including or excluding a specific list of words. Unfortunately, it does not offer any way to take the dispersion into account. However, it offers to extract “key n-grams” which are n-grams that appear in one (sub-)corpus and not in another or n-grams that appear in both (sub-)corpora.

3 Available at <https://app.sketchengine.eu/>

Figure 2: Sketch Engine's nested n-grams

	Word	Frequency [?]
1	As a result	2,363 ...
2	At the end	1,623 ...
3	At the end of	1,510 ...
4	• the end of	20,719 ...
5	• At the end	1,623 ...
6	At the same	2,182 ...
7	At the same time	2,108 ...
8	• the same time	7,564 ...
9	• At the same	2,182 ...

1.4 Conclusion

As demonstrated through this chapter, the recurrent sequences studied with the cluster analysis methodology are substantially varied, and many parameters can be taken into account to identify and filter them. Unfortunately, the tools available to non-computational linguists do not reflect this diversity. The most important feature lacking is a good handling of nested n-grams. The only tool that takes nested n-grams into account is the Sketch Engine in which, if an n-gram has recurrent sub-n-grams, they are listed below it. However, it can be more relevant to see all the super-n-grams of a given n-gram written under it rather than its sub-n-grams: while sub-n-grams can easily be inferred – they are literally written inside the n-gram being considered –, super-n-grams are harder to guess. For example, the n-gram “at the same” has exactly two sub-n-grams, namely “at the” and “the same”, but to know all its super-n-grams it would be necessary to check all the n-grams that start or end with “at the same”, and there might be many.

Available tools do not either handle well p-frames: only KfNgram and WordSmith Tools take them into account but, in the former, only one p-frame size can be extracted at a time and, in the latter, p-frame variants are simply written in grey and struck through, but they are not written next to their corresponding p-frame.

Finally, no tool offers to include POS tags in the MWUs, for example to fill the free slot of p-frames.

2 Co-occurrence analysis

In this section, I will present the second method to automatically extract MWUs: the co-occurrence analysis. This section, like the previous one, focusses on how to extract MWUs, and not how to analyse them. First, a definition and a general methodology will be presented (2.1), followed by a more detailed explanation of the technical aspects of the methodology (2.2). After that, I will present some available tools to help conduct such analyses (2.3).

2.1 Definition and methodology

Co-occurrence analysis draws attention to groups of words that appear together more often than they would by chance. For example, next to the word *tea*, we often find *cup*, *spoon*, *brew* or *drink*, but we do not expect to find next to it, other than by chance, words such as *computer*, *applaud* or *sky*. The concept of “chance”, which might sound abstract in the given definition, becomes more concrete when interpreted numerically thanks to statistics and probabilities. For that purpose, association measures (AMs) were developed to compute how strongly a group of words is associated. If the association of a group is stronger than a given threshold, it means that the words constituting this group appear together more often than they would by chance and they can be considered as a MWU. Most AMs use statistics based on the frequency of the words appearing together or without each other, but other criteria can also be used such as conditional probabilities or linguistic features. AMs will be described more fully in the section 2.2.2.

Unlike cluster analysis, co-occurrence analysis usually treats words together regardless of their order in the sentence and allowing gaps of several words between them. The resulting MWUs, such as *give* + *presentation* or *heavy* + *traffic*, are usually called collocations. The associations can theoretically be measured any linguistic units (characters, morphemes, tokens, lemmas, POS tags, etc.), but in the vast majority of time, to identify MWUs, they are used on tokens or lemmas. Working with lemmas and not considering the order in which the words appear highlights very flexible MWUs. For example, the single the collocation *give* + *presentation* can have different realisations, such as illustrated in examples (1) to (3).

- (1) gives a presentation
- (2) gave presentations
- (3) a presentation was given

The main steps to extract MWUs in the co-occurrence analysis can be summarized as follows:

Step 0 Segment (and lemmatise) the corpus

Step 1 Consider in turn each word of the corpus (= the node) and count all the words (tokens or lemmas) co-occurring with it in a window of n words (node + co-occurring words = MWU candidates)

Step 2 Use an AM to compute how strongly the words of each candidate are associated

Step 3 Filter the pairs to keep the most relevant ones (= MWUs)

In Step 0, the corpus is pre-processed, if it was not already done. For the co-occurrence analysis, the corpus is usually lemmatised and the analysis is conducted on lemmas and not on tokens. This is done to group together all the possible syntactical realisations of a collocation, as illustrated in the examples (1) to (3).

Step 1 is illustrated in Figure 3 on the famous Shakespeare quotation (*Hamlet*, Act 3, Scene 1). In this example, all pairs of words co-occurring in a window of 2 words to the left and right of the node are linked by a line. Since pairs of words were selected here, all our candidates will be made of two words. The order in which the words appear does not matter (*or + to* is counted together with *to + or*).

Figure 3: Illustration of the candidate selection with a window of 2 words, each candidate containing 2 words



The Step 2 is at the heart of the co-occurrence analysis: it determines the method to compute how strongly two or more words are associated. The first researchers who used AMs to identify MWUs are Church and his colleagues (1989). Their idea is based on the psycholinguistic definition of the term *word association*: two words show an association if they are processed quicker by a subject when they appear together; for example, *nurse* would be processed faster if it occurs after the word *doctor*. Church et al. (1989) had the idea to use statistics to transpose this psychological concept onto corpus data. The notion of co-occurrence was mainly popularised by John Sinclair (1991, chapter 8) and many researchers have developed AMs aiming at identifying the best word associations.

Most of the AMs are designed to compute the association between two words. As a result, the co-occurrence analysis focuses most often on pair of words. Few AMs are

designed to measure the association of more than two words, but there exist techniques to extend binary AMs to longer groups of words. Section 2.2.4 is dedicated to that issue.

Step 3 is quite explicit and is partly similar to what was explained for the cluster analysis. The aim is to filter the candidates to keep only the relevant ones as MWUs.

These steps are a generalisation of the most commonly used methodology in the co-occurrence analysis, and the results vary according to how these steps are operationalised. The different possible implementations and their impact on the results are explained in the following section.

2.2 Implementation

Each of the three steps presented can be operationalised in different ways to fit to a specific purpose. The three steps (candidate selection, AMs, filters) will be discussed and exemplified in this section.

2.2.1. *Candidates selection*

The purpose of making a MWU candidate selection in the co-occurrence analysis is to identify all groups of words that could potentially be associated, discarding the others. There are two reasons to do this. First, by discarding the groups of words that are not a priori associated, we reduce the number of false positive results, i.e. groups of words that would be identified as MWUs while they appear together only by chance. For example, in (4), *presentation* and *gave* influence the use of each other but, in (5), they appear together only by chance; in (4), *presentation* and *gave* form a phraseological expression, and, as such, we want to select them among our candidates, whereas, in (5), they appear together by chance and should either not be selected among the candidates, or be filtered out in the last step of the automatic extraction – in the given example, an easy way to filter it out would be to discard any groups of words that are separated by a punctuation mark. Ideally, all the candidates should consist of words that are syntactically related. For that reason, several authors recommend to use a parser to do the candidate selection (Watrin, Patrick 2007; Nerima et al. 2006). The second advantage of making a candidate selection is of a computational nature: AMs will be applied on each MWU candidate, which means that the amount of computation needed increases linearly with the number of candidates. Conversely, limiting the number of candidates decreases the computational complexity. In a nutshell, the aim of the candidate selection is to select the highest number of relevant candidates while keeping this amount low enough to be manipulable by the computers at our disposal.

(4) The presentation they gave was very clear.

(5) Through their presentation, they gave examples of possible applications.

The most common approach to the candidate selection is the one that was presented in the general methodology: select all groups of words occurring together in a **window of n words**. In that case, the extracted MWUs can be referred to as “positional collocations” (Evert 2004). This method is favoured due to its simple implementation process. The idea behind it is that words that are syntactically related to each other usually occur near each other. Experimental evidence has shown that, in English, most lexically associated words are separated by at most five words (Smadja 1993; Dias et al. 2000). Researchers usually select a window size between 3 and 5 words to the left and right of a given word, called the *node*. Depending on the kind of phraseological expressions that interest researchers, they can decide to count together words regardless of their order (*give ... presentation* will be counted together with *presentation ... give*) or to count them separately. The former method will identify less fixed MWUs than the latter.

One disadvantage of using a window to select MWU candidates is that many groups of words are selected although they are not syntactically related, and therefore most probably happen to be next to each other by chance, as illustrated in example (5). This results in a long list of candidates, most of which do not constitute a MWU (they will not, or at least should not, be identified as such by the AM in the second step). To reduce the number of candidates and select them more accurately, some authors do not use a window of words, but instead select, with the help of a parser, all groups of words that are syntactically close. This time, the extracted MWUs can be called “relational collocations” (Evert 2004). Nerima et al. (2006) and Evert et al. (2017) have shown that this significantly improved the results by including pairs of words that appear far from each other and yet are **syntactically related**. They argue that it is even most useful in languages with a word order more flexible than English, such as German or Russian. The downside of this method is that it requires a parsed corpora; manual parsing takes time, and automatic methods are not always accurate, especially with less standard data, such as spoken language.

For some studies (e.g. Joseph 2013; Fazly & Stevenson 2006), instead of the methods already presented, researchers select as candidates all groups of words corresponding to some **parts of speech patterns**, such as verb-noun or adjective-noun associations. This greatly reduces the number of candidates and it is a way to focus in more detail on a specific kind of MWUs. This method is especially used with AMs that take into account linguistic variation (more detail on linguistic AMs is given in section 2.2.2).

As for the length of the MWU candidates, the most usual one in this methodology is two words, probably because of the influence of Sinclair (1991). Therefore, many binary association measures available, which highlight common associations between, for example, a verb and a noun, a verb and a preposition or an adjective and a noun. Although three- and four-words AMs are available (see Pedersen et al. 2011), to the best

of my knowledge, no linguistic study used them. There also exist some AMs that are designed to work on several sizes of candidates. With such AMs, it is possible to select candidates of several sizes at the same time, and to filter out the less relevant candidates in Step 3 – usually discarding shorter candidates if a longer one is more pertinent. This will be further explained when describing multi-word AMs. There also exist techniques to extend binary associations to multi-word associations that will be explained in section 2.2.4; in this particular case, it is not necessary to set a specific number of words in advance to constitute the candidates.

2.2.2. Association measures

There is a considerable number of AMs and a large body of literature compares different measures (e.g. Evert et al. 2017; Pecina 2005; Wiechmann 2008; Kolesnikova 2016). In short, AMs can be bidirectional or directional, designed for two word associations or multi-word associations, based on statistics, probabilities or on the resistance to linguistic transformations. This section presents the main AMs that correspond to those different features.

Bidirectional or symmetric AMs are based on contingency tables, like the one presented in table 2. In such a table, we can read at a glance the number of occurrences of two words together (the joint frequency, cell a) or without each other (the marginal frequencies, cells b and c). A common way to compute a bidirectional AM is to compare this table of observed frequencies with a hypothetical table containing the same cells, with the expected frequencies in case the two words were randomly distributed. The latter table represents the hypothesis of the two words being independent from each other, and statistical tests can measure if, and how strongly, the former table differs from that hypothesis. If it differs significantly enough, this means that the two words are used together more often than they would if they were used randomly. The result of the computation of such AMs reflects the *mutual association* of the two words, i.e. how strongly they attract each other.

Table 2: Contingency table of two words

	$word_1$	$\neg word_1$	Total
$word_2$	$a = freq(word_1 word_2)$	$b = freq(\neg word_1 word_2)$	$a + b$
$\neg word_2$	$c = freq(word_1 \neg word_2)$	$d = freq(\neg word_1 \neg word_2)$	$c + d$
Total	$a + c$	$b + d$	$a + b + c + d$

Among the most used bidirectional AMs are the log-likelihood test (Dunning 1993), the t-score (Church et al. 1991) and the mutual information (MI, Church et al. 1989)). Since

those three measures are based on the same underlying principle, they are globally well correlated (see Evert et al. (2017) and their interactive graph⁴ that displays the precision and recall of different AMs depending on different extraction parameters). Even so, they each highlight slightly different kinds of MWUs. The MI gives a higher score to groups that contain very infrequent words, like *chewing gum* or *post mortem*. In contrast, the t-score and the log-likelihood test favour frequent words and will identify *can be* or *part of* as strong collocations. The log likelihood works better with fewer data than the t-score. It is possible to combine several statistical AMs to take advantage of their specificities; Pecina & Schlesinger (2006) have demonstrated that combining several AM yields better results, however, Gries & Durrant (2021) argue that this choice is usually a pragmatic decision rather than a theoretically motivated one and they advice to avoid mixing AMS.

As explained above, bidirectional AMs quantify the *mutual association* of two words: they don't make a difference if *Pyrrhic* attracts *victory* or if *victory* attracts *Pyrrhic*. As already underlined by Kjellmer (1991) and Sinclair (1991), asymmetry is an important property of collocations and it should be taken into account by AMs. To answer that limitation, some researchers developed **directional or asymmetric AMs** (Pedersen 1998; Michelbacher et al. 2007; Gries 2013). Directional AMs are based on conditional probabilities, i.e. on the probability of the word w_1 to occur, knowing that the word w_2 occurred. The most widely used directional measure is ΔP (Gries 2013; Ellis 2006) defined in the equation 2.6. It is equal to the probability of the word w_1 to occur knowing that the word w_2 occurred, minus the probability of w_1 to occur knowing that w_2 did not occur. If $p(w_1|w_2)$ and $p(w_1|\neg w_2)$ are close, it means that the likelihood of observing w_1 is not influenced by the use of w_2 and ΔP is close to 0. As ΔP approaches 1, it reflects the fact that observing w_2 increases the chances of observing w_1 . Inversely, as ΔP approaches -1, it means that observing w_2 decreases the chance of observing w_1 . The result obtained will thus not be the same if w_1 and w_2 are inverted.

$$\Delta P = p(w_1|w_2) - p(w_1|\neg w_2) \quad (\text{Eq. 2.6})$$

Directional AMs have been shown to correlate with results obtained in psycholinguistic studies. For example, Michelbacher et al. (2007) found similar associations between the directional word associations found in a corpus and the associations made by people when they have to find words related to a given word. As for Ellis (2006), he could explain the difficulties of acquiring a second language with the help of ΔP and associative learning theories. Thanks to such studies, directional AMs are based on a more solid theoretical background than bidirectional AMs. Used alone, they might not be as effective as bidirectional AMs in identifying phraseological expressions, but they

4 http://corpora.linguistik.uni-erlangen.de/projects/evaluation/elex_2017/ (consulted on 3rd Feb. 2021)

provide more information about the identified MWUs by making explicit the direction of the association.

Multi-word AMs. The AMs presented so far are designed to measure the association between two words. However, phraseological expressions are not limited to two words associations; there is a need for AMs that measure the association between any number of words. Some researchers have adapted binary AMs to groups of more than two words, with always the same underlying idea: a binary AM is used to measure the strength of association between the words of the groups two by two and then all the results obtained are combined. For bidirectional AMs, Banerjee & Pedersen (2003) extended the MI for groups of three words and the log likelihood measure for groups of three and four words, but I am not aware of any study that used those measures. For directional measures, Dunn (2018) developed several extensions of ΔP such as mean ΔP , sum ΔP and min ΔP . By combining several of those measures, the author finds highly relevant MWUs, while filtering out non-relevant ones. He demonstrates once more that the direction of an association is an important feature of word combinations; in multi-word associations, the direction of the association between adjacent pairs of words can help differentiate between true phraseological expressions (e.g. *the european union*) and words that appear together often by chance (e.g. *of the european union and*). Next to those adaptations of binary AMs, some researchers have specifically designed AMs to compute the association of groups of more than two words. The AM of this kind that has shown the best results so far is the mutual expectation (ME, Dias 2003). The ME, defined in Eq. 2.7, is the product of the probability of a candidate times the normalized expectation. The normalized expectation (NE) is illustrated in Eq. 2.8 for a group of four words; the principle would be the same for any number of words. This formula measures how strongly a group of words resists the loss of any of its constituents. Using an algorithm based on the ME, Dias (2003) could identify phraseological expressions such as compound nouns and verbal, adjectival and adverbial locutions.

$$ME(candidate) = p(candidate) * NE(candidate) \quad (\text{Eq. 2.7})$$

where $p(candidate)$ is the relative frequency of a candidate.

$$NE(ABCD) = \frac{k(ABCD)}{\frac{1}{4}(k(ABC) + k(ABD) + k(ACD) + k(BCD))} \quad (\text{Eq. 2.8})$$

where $k(w)$ is the number of occurrences of w and A, B, C and D are words that constitute a candidate.

In addition to those AMs based on statistics and probabilities, there exist **AMs based on linguistic information** that test to what extent groups of words (e.g. *spill the beans*) resist linguistic transformations such as passivisation (*the beans were spilled*), insertion (*spill the official beans*), substitution (**spill some beans*) or a change in number (**spill the bean*). Such AMs are based on the idea that phraseological expressions are more

fixed than compositional phrases and usually appear in the same form. This is true to some extent, but it is not an absolute rule, and the different transformations that an expression can undergo cannot be predicted (Sag et al. 2002).

To compute such measures, the candidates must correspond to only a specific kind of MWU, such as noun-verb associations. This is required because all the candidates must potentially undergo the same linguistic transformations (it would not be relevant to take into account, for example, that an adjective-noun association cannot undergo passivisation). The idea is to compare, in a corpus, the different occurrences of each MWU candidate. If the words constituting the candidates always appear in the same form, or with very little variation, they are idiomatic; if they show a lot of internal variation, they are more probably compositional. Fazly & Stevenson (2006), who focused on verb-noun idiomatic constructions, obtained good results at differentiating idiomatic from compositional phrases. Their system uses statistic measures to compare a candidate to its variants, taking into account the syntactic as well as the lexical fixedness. To compute the lexical fixedness, they compare the MI of a candidate with its lexical variants, each variant containing a synonym of the original words of the candidates (e.g. *kick the habit* as candidate and *boot the habit* or *kick the custom* as variants). If the candidate has a MI much higher than its synonyms, it means that it is lexically fixed. As for the syntactic fixedness, the syntactic behaviour of the candidate, i.e. the linguistic transformations observed on the candidate, is compared to the typical behaviour of verb-noun pairs; this highlights candidates that show more syntactic fixedness than average. The authors finally used a measure that combines the lexical and syntactic fixedness and obtained good results on automatic extraction tests with 74% of accuracy and 48% of recall, with better results obtained when focussing on the identification of more frequent MWUs. Joseph (2013) made a similar study on French and compared her results with several bidirectional AMs. Her linguistic AM showed better results than the bidirectional AM, especially when less data is available.

Needless to say, linguistic AMs are harder to implement than purely statistical approaches since they require language specific linguistic expertise. Despite this difficulty, more research should be devoted to this kind of AM because they offer very promising results: they require less data to provide good results, they offer a new insight on word associations, and they can provide information on the kind of variation that each MWU can undergo. This kind of information is needed both for lexicographers and for some NLP tasks, such as language generation, to know how to use each MWU. Finding that information automatically would substantially help in those two domains.

In conclusion, each kind of AM has its advantages and disadvantages. The bidirectional AMs are easy to implement and produce quite good results, but they require lots of data and are usually not based on strong theoretical linguistic background. The directional AMs are equally easy to implement and provide information about the direction of the

association, which reflects psychological mechanisms in the processing of language. Multi-word AMs open the door to studying the associations of more than two words, which reflects the reality of phraseological expressions. The linguistic AMs use a linguistic definition of phraseology to identify MWUs, they are harder to implement, language-specific and designed for a specific kind of MWUs (e.g. noun-verb associations), but they require less data than statistical AMs to yield good results. They also provide an insight on the variability allowed by each MWU. All those measures are continuous, which highlights that the difference between compositional phrases and phraseological expressions is not a dichotomy but a continuum.

2.2.3. *Filters*

In the co-occurrence analysis, the main filter of the candidates is to set an association threshold, that is, a degree of association under which the candidates are not considered to be MWUs. However, other filters are possible to refine the results and reduce the number of false positives, such as linguistic filters, e.g. syntactic patterns and stop-words, and dispersion. Those different filters will be briefly presented here.

Association threshold. As explained above, the AMs quantify the strength of the association between the words constituting a candidate. With that information, we can set a threshold to decide how stringly associated the candidates must be in order to consider them as MWUs, i.e. to consider that they occur together more often than they would by chance. The value of the threshold must be adapted depending on the AM(s) chosen and on the size of the corpus. If corpora of different sizes are used, the threshold must be adapted to each corpus size, and the results from the different corpora should be compared with caution, since the value of the AM will not correspond from one corpus size to the other. Note that if candidates of several sizes are selected, it is likely that some candidates are comprised in longer ones, like it was the case in the cluster analysis. In that situation, the candidates selected as MWUs are usually those that show a stronger association than all longer candidates they are part of, as well as all the candidates that are comprised inside them. For example, in (6), we would expect (6)(c) to be selected and all the other candidates to be discarded.

(6) *Fictive example of nested candidates with their association strength.*

- (a) the european: 0.7
- (b) european union: 0.7
- (c) the european union: 0.8
- (d) of the european union: 0.5
- (e) for the european union and: 0.4

As presented in the section dedicated to the candidate selection (section 2.2.1), it is possible to select candidates following a certain **syntactic pattern** and thus limit the results to one kind of sequence. Smadja (1993), however, did not want to limit the results and that way and decided instead to use the syntax as the last filter of his

extraction methodology. After applying AMs on all pairs of words co-occurring in a window of 5 words to the left and the right of the nodes, he used a parser to check the syntactical relation of each occurrence of the candidates. If the same syntactical pattern (e.g. verb-object) was found in (almost) each occurrence, the candidate was kept; if no such pattern was found, the candidate was rejected. In his experimentation, Smadja (1993) filtered out around half of the candidates in this way, including, for example, *savings* + *continue* and *securities* + *dealer*. However, it seems that, in more recent studies, parsers are usually used for the selection of candidates rather than to filter them (e.g. Inurrieta et al. 2020; Duan et al. 2020).

Finally, like in cluster-analysis, it is possible to filter the results with a list of **stop-words**. In the co-occurrence analysis, punctuation marks are never relevant as part of the association – whereas a sequence such as “, however, “ can be useful in cluster analysis, for example to observe that the word “however” is often used between commas, it does not seem relevant to say that “,” and “however” are strongly associated, because this does not provide information on how to use that punctuation mark next to that word. As in co-occurrence analysis, tokens as well as lemmas and POS tags can be set as stop-words.

2.2.4. *An extension of the co-occurrence analysis: recursive algorithms*

The general methodology presented so far in this section, except for the ME and the ΔP extensions, are designed to work on a specific number of words, which limits the scope of studies of MWUs to only one size. To overcome this limitation, some authors have developed recursive algorithms, that have the advantage of extracting various size MWUs without any nested ones. The idea behind using recursive algorithms to extract MWUs is to compare (groups of) words two by two and to construct multi-word units from binary associations. Two main algorithms were developed on this principle: the algorithm MERGE, developed by Wahl & Gries (2020) and based on Wible et al. (2006) and the algorithm implemented in the software ConcGram (Cheng et al. 2006). The former is presented here, while the latter is presented in the Available tools section.

MERGE is the name given by Wahl & Gries (2020) to their algorithm which is greatly inspired from the one developed by Wible et al. (2006). Those two algorithms can be summarized in four steps:

Step 1 Extract candidates (pairs of tokens) from a corpus

Step 2 Compute a binary AM on all the candidates

Step 3 Merge the pair that show the strongest association (that pair now count as a single token)

Step 4 Iterate steps 1 and 4 until a stop condition is met

The first two steps are similar to a traditional extraction of co-occurrences and the discussion conducted in the sections 2.2.1 and 2.2.2 apply here too. In the case of Wahl & Gries, they leave the choice of the size of the window to the user and use the log-likelihood as AM, whereas Wible et al. choose a window of five words and use both the MI and a conditional probability measure. In the Step 3, the tokens that are the most strongly associated are “merged”, that is, they form a MWU and are considered from now on as a single token in the corpus. At this point, the corpus contains different tokens since some of the original tokens were merged. The steps 1 to 3 can thus be reapplied on the (modified) corpus and will yield different results. From the second iteration, some tokens of the pairs (MWU candidates) can be made of two words that were merged during a previous iteration. For example, in the first iteration, *matter* and *fact* can show the highest association in the corpus and thus be merged. At the next iteration, *matter* and *fact* are considered together as one token: *matter + fact*. When the candidates are selected, they can thus be made of *matter + fact* and another token, say, *of*. If *matter + fact* is strongly associated with *of*, those two tokens (three orthographic words) will be merged, forming the MWU *matter + of + fact*. By repeating those steps, increasingly longer MWUs are formed while always computing binary associations. The process is repeated until a stop condition is met. For Wible et al., the stop condition is that no candidate matches a given threshold for both AMs (MI and conditional probability measure). For Wahl & Gries, two stop conditions are possible: either until no candidate matches a given association threshold or until a number of iterations is accomplished. Note that, unlike Wahl & Gries, Wible et al. do not allow for positional variation (*of ... fact* is counted separately from *fact... of*).

The version of the algorithm implemented by Wible et al. highlights in the corpus in different colours the MWUs identified with the MI value, the conditional probability measure or with both. Wahl & Gries’ algorithm outputs both a list of the identified MWUs and the original corpus annotated with the MWUs.

This method allows to identify MWUs of different lengths, without identifying shorter units that would be comprised in longer ones (if *matter + of + fact* is identified, *matter + fact* cannot be). Furthermore, unlike the cluster analysis, recursive algorithms allow for words to appear in a different order and apart from each other, as long as they occur together in a given window of words.

2.3 Available tools

Most corpus analysis tools provide a set of binary AMs that can be used on all pairs of words in a given window or on all words appearing near a given word. It is the case, for example, of the English-Corpora interface (Davies 2016-present) and of all the tools that were presented in the section dedicated to the cluster analysis. Although these tools are useful in identifying collocations, they do not offer any flexibility: they only provide

a predefined set of AMs and do not offer to filter the results; this must be done manually. Some other tools were designed specifically to help use AMs on corpora. Those more specialised tools are the focus of this section.

To my knowledge, the **Ngram Statistic Package** (NSP, Banerjee & Pedersen 2003) is the first software designed for that purpose. It consists of two main command-line tools written in Perl. The first tool, `count.pl`, counts the joint and marginal frequencies of each pair of words occurring together in a certain window. The output is given in a text file, each line corresponding to one group of words along with all the relevant frequencies. The second tool, `statistic.pl`, allows to apply AMs based on the frequencies extracted by `count.pl`. NSP implements around ten AMs to compute associations from two to four words. The program enables the user to configure the pre-processing of the corpora given as input and, in particular, to implement the tokenisation process. It also offers the possibility to set stop-words. Thanks to the computation in two steps, an advanced user can use the information provided by `count.pl` to compute his own AMs. Unfortunately, NSP does not offer to lemmatise the text, so that it works only on tokens (unless the corpus is lemmatised by the user before using NSP). Other basic tools are provided with NSP to facilitate the manipulation of the data (sorting the results, combining results from several corpora, etc.)

The **UCS toolkit** (Evert 2004-2010) is another command-line tool that can apply AMs on corpus data. It is written in Perl and R and implements more than 40 binary AMs. It can either take a count file produced by NSP or a raw text as input. Like the previous one, this program leaves the possibility to advanced users to implement their own AMs. It also allows to make complex queries on the data to work only on specific pairs, corresponding to given words or frequencies. The outputs are stored in a text tabular format, with each row corresponding to a pair type. It also implements a procedure for evaluating the performance of AMs. This tool is in general more powerful and flexible than NSP, but it is limited to two word associations.

The **mwetoolkit** (Ramisch 2012; Ramisch 2020) is a command-line tool that allows to make complex queries on tagged and parsed corpora. The queries are similar to regular expressions, and can include word forms, tags and syntactic relations. It can either extract a list of MWUs matching the patterns or tag them in the corpus. Once the MWU candidates are extracted, it is possible to compute their degree of association with one of five AMs. This is the only currently available tool that offers such powerful queries, taking into account both syntactic and statistical information. Its downside is that, because of the many features, the program is divided into many sub-programs that must be used one after the other. This makes this command-line tool quite hard to use.

Finally, the software **ConcGram** (Cheng et al. 2006) implements a recursive algorithm that can be used to extract nested MWUs, namely *concgrams*. A concgram is defined as

“all the permutations of constituency variation and positional variation generated by the association of two or more words.” (2006: 414). This means that concgrams are MWUs that allow for discontinuity and for changes in the word order. As an example, the concgram *Asia/world/city* can be realized, among others, as *world city of Asia* or *Asia’s world city*. The automatic identification of concgrams in the software ConcGram works as follows:

Step 1 List all unique words in the corpus

Step 2 Search for all the concordances of each word (the “origin”) in a user-defined window

Step 3 List all words co-occurring with the origin (= concgrams)

Step 4 If the user wants it, the list of concgrams output at Step 3 can serve as input to Step 2 to reiterate the algorithm

In Step 4, the user can perform the same search, taking consecutively 2-word, 3-word and 4-word concgrams as the origin to identify respectively 3-word, 4-word and 5-word concgrams. The software does not allow concgrams longer than 5 words because the authors argue that the results would not be pertinent to find phraseological expressions in English. As the reader may have noticed, no AM needs to be used in the extraction of concgrams, but the t-score and MI value can be used to filter the results with minimum threshold values.

The authors intended the program to help identify the “aboutness” of a text and it achieves this purpose well (Cheng 2009; Greaves & Warren 2007), but it requires a lot of manual vetting when it is used to identify phraseological expressions. However, ConcGram allows to select a specific word to focus on the concgrams made up with that word. This feature differs from what is proposed by the other software tools presented and this can be used to have a different approach on the data.

2.4 Conclusion

In conclusion, purely statistical methods have the benefit of being easy to implement, to require little linguistic expertise and to be applicable to different languages with no change in the implementation. Among statistical methods, recursive algorithms have the advantage of identifying MWUs of any length, without any shorter unit nested in longer ones. Linguistic AMs are harder to implement and must be adapted for each language, but they are much more sensitive to phraseological expressions, even when few data are available. Since linguistic AMs are harder to implement, a compromise is to use statistical AMs with linguistic filters, for example to ensure that all words in the candidates are syntactically related.

The mwetoolkit is by far the most advanced tool to extract MWUs following the cluster analysis method. It is highly customisable, and offers to take into account both

statistical and linguistic information. Its main limitation, as highlighted by the creator himself, is the use of terminal commands, which might discourage users that are not used to typing commands in a Unix prompt (Ramisch 2020). However, like all other tools designed to automatically extract MWUs, it does not implement any directional AMs, which makes them accessible only to computational linguists who are able to implement them themselves.

With regard to recursive algorithms, it is unfortunate that an implementation of MERGE is (at least currently) not publicly available. A disadvantage of ConcGram compared to MERGE is that it outputs a different list for each size of MWU, which means that the problem of nested MWUs remains.

3 Conclusion

The diversity of phraseological expressions is reflected in the diversity of implementations to extract MWUs. This chapter detailed the different implementations of two main methodologies: the cluster analysis, that focuses on frequent continuous sequences, and co-occurrence analysis, that focuses on groups of words that occur often near each other. The prototypical implementation of each method is summarized in Table 3.

Table 3: Comparative table of cluster analysis and co-occurrence analysis

Cluster analysis	Co-occurrence analysis
• Focus on frequency • Fixed word order • Continuous sequences • Between 3 and 6 words	• Focus on association strength • Varying word order • Groups of words • 2 words

For both methods, many parameters such as the frequency, the association strength, the dispersion and the linguistic patterns can be used to extract different kinds of MWUs. Automatic phraseology extraction tools should reflect this diversity and allow for a great customisation in the extraction process to fit the different needs and purposes of the users. To this day, even though many tools exist (the most common were presented above), very few offer to customise the extraction process, and this limits the scope of studies on phraseology to what is feasible with those tools. To summarise, the challenges that must be addressed by automatic phraseology extraction tools are (1) to allow for the extraction of MWUs of several sizes while filtering out nested sequences, (2) to take into account all the aspects that are part of the definitions of phraseology,

such as the frequency, the word association strength, the dispersion and the linguistic variation, (3) to integrate different levels of precision by extracting MWUs that include part of speech tags, lemmas and/or tokens, (4) to make the results as accurate as possible to reduce the manual vetting.

The next chapter presents a tool, the Phraseology Extraction Package, that was designed to tackle some of those challenges.

Chapter 3

Phraseology Extraction Package

Given the limitations mentioned in the state of the art, there is a need for flexible tools to automatically extract word combinations from corpora, and in particular to handle nested MWUs. I have been developing the Phraseology Extraction Package (PEP) to answer that limitation. PEP takes the form of a python package, usable in command-line. The features of this tool are: (1) the extraction process is highly customizable, (2) n-grams of several lengths can be extracted while handling nested n-grams, (3) the resulting list of MWUs is displayed in a hierarchical structure that groups together similar MWUs, (4) both fixed and less fixed MWUs are extracted by taking into account not only tokens, but also lemmas and part of speech tags.

In this chapter, I will first describe the implementation of PEP, then I will illustrate how to use PEP and discuss the results. Finally, I will present the limitations of PEP and possible future improvements.

1 Implementation

The source code of PEP and its documentation are available online at <https://github.com/Reyla98/PhraseologyExtractionPackage>. The version described and used in this thesis is the version 0.2.0, released in May 2021. PEP is being developed as a python package and is available on all platforms (Windows, MacOS, Linux). The extraction of MWUs can be launched with command lines from a terminal; there is currently no graphic interface.

First, this section first gives an explanation of the general principles underlying PEP's MWU extraction. Then, a more thorough description of each step of the extraction is given. Finally, some examples of MWU extractions will be given.

1.1 The ideas behind PEP

Through the State of the Art chapter, it was illustrated that MWUs can take many forms: they can be of different lengths and be constituted by different linguistic elements such as tokens, lemmas or POS tags. The available tools poorly handle variation in MWUs: they handle poorly MWUs of different lengths and usually do not integrate other elements than tokens or lemmas in the MWUs. Filling this gap has been the main objective of the MWU extraction process in PEP.

This section starts with a presentation of how PEP selects MWU candidates, followed by a description of the treatment made to nested candidates. Subsequently, it is explained how PEP builds MWUs, then how they are filtered, and finally how they are displayed.

1.1.1. *Extracting MWU candidates*

To fit to the objective of integrating several kinds of linguistic elements into the MWUs, it was necessary to have access to a lemmatised and tagged corpus. Since many corpora are neither lemmatised nor tagged, PEP uses TreeTagger (Schmid 1995) to perform that processing.

To fulfil the objective of extracting MWUs of different lengths, the starting point of the extraction is to list all the n-grams found in the corpus that would match the different lengths specified by the user. As mentioned in the previous chapter, an n-gram is a sequence of n words and when n-grams of several lengths are extracted at the same time, some n-grams are inevitably nested in longer ones. This is illustrated by example (7) where “at the same” is nested in “at the same time” and “at the same place”. To refer to that phenomenon, I will adopt a terminology close to the one introduced by Dias (2002): I will name *super-n-grams of an n-gram* all the longer n-grams in which a shorter one appears and *sub-n-grams of an n-gram* all the n-grams that are nested in one. In example (7), (7)(a) is sub-n-gram of (7)(b) and (7)(c). Inversely, (7)(b) and (7)(c) are the super-n-grams of (7)(a). Together, they form a *family*.

(7) *Examples of nested n-grams.*

- (a) at the same
- (b) at the same time
- (c) at the same place

To keep the final list of MWUs relevant, PEP does two treatments to nested n-grams: some nested n-grams are discarded and the remaining n-grams are grouped into

“families”, in a hierarchical structure, and transformed into *patterns*. Those treatments are explained respectively in sections 1.1.2 and 1.1.3.

1.1.2. *Filtering out irrelevant nested n-grams*

Let’s consider the following (made-up) list of n-grams where (8)(a) is nested in (8)(b), and (8)(c) is nested in (8)(d) and (8)(e); the numbers written at the right of the n-grams represent their frequency:

(8) *Example of nested n-grams.*

- (a) on the one - 10
- (b) on the one hand - 10
- (c) at the same - 16
- (d) at the same time - 12
- (e) at the same place - 4

In this list, we can make two observations. First, “on the one” and “on the one hand” both occurs 10 times. This means that “on the one” is *always* followed by “hand”. Second, we see that “at the same” appears 16 times in total and has two variants: “at the same time” and “at the same place”, which occur respectively 12 and 4 times.

As did Merkel & Andersson (2000), PEP infers from that data that “on the one” is not a good MWU candidate because it is not a complete unit; since it is always followed by the same word, that word should be part of the MWU. For that reason, all sub-n-grams that have the same frequency as their super-n-gram (in that case there can be only one super-n-gram) are discarded. However, since “at the same” has two super-n-grams, it could be a relevant MWU, as well as its super-n-grams. In that case, PEP keeps both the sub-n-gram and the super-n-grams as MWU candidates.

This choice of keeping all the n-grams of a family that have different frequencies instead of keeping only the most frequent one is reinforced by the statement made by (Wahl & Gries 2020):

However, the fact that lower-order n-grams are entirely discarded if a higher-order n-gram containing them is stronger is potentially problematic: while certain tokens of a lower-order n-gram may be fragmentary (*fingers crossed* in to keep one’s fingers crossed), others may not be (*fingers crossed* in Speaker A: “I hope we win!” Speaker B: “Fingers crossed!”).

When they say “stronger”, they are referring to the strength of association between two or more words, measured by an association measure, but this applies for frequency as well; a longer n-gram is always at most as frequent than as its sub-n-grams, being less frequent does not mean less phraseological. Since frequency alone is not enough to tell

if a sequence is phraseological or not, at this step, PEP keeps all candidates, also the less frequent ones.

1.1.3. *Building patterns*

The remaining n-grams still contain many nested n-grams. Those n-grams cannot simply be printed in decreasing order of frequency nor in alphabetical order because, in those cases, the resulting list would be hard to interpret since related (nested) n-grams would appear apart from each other, as illustrated in example (9). It is much more relevant to group n-grams in “families” as proposed by Lefer & Grabar (2015) and illustrated in example (10).

(9) *Nested lemma n-grams sorted by alphabetical order.*

can not remember
do not know
I can not remember
I do not know
we do not know
you can not remember

(10) *Nested lemma n-grams grouped in families.*

- (a) can not remember
 I can not remember
 you can not remember
(b) do not know
 I do not know
 we do not know

PEP groups in a family all n-grams that share a common core of lemmas. That core corresponds to the smallest lemma n-gram in the family. In example (10), the core of the family (a) is “can not remember” and the core of the family (b) is “do not know”. As a consequence, the common core is always at least as big as the minimum length of n-grams specified by the user for the extraction (e.g. if the user wants to extract n-grams from 3 to 5 words, the common core would be constituted of at least 3 lemmas).

PEP does not only group together n-grams belonging to the same family, it also builds *patterns* from those n-grams. In PEP, a pattern is a MWU comprised of tokens, lemmas, POS tags or wildcards. To build a pattern, all n-grams of a family are compared word by word and, in each case, the most specific common element is selected to constitute the pattern. To illustrate that, let’s consider the n-grams in example (11) (note that the n-grams treated by PEP are lemmatised and tagged, but to improve the readability of the example, only tokens are transcribed). All those n-grams share the common lemma core “I do not know”, which corresponds to the lemma n-gram of the smallest n-grams in the list (namely (11)(a) and (11)(b)), they thus belong to the same family. PEP aligns those n-grams so that the common core always appears in the same position. In this case, “I”, “do”, “not” and “know” appear respectively in position 2, 3, 4 and 5 since there is at

most 1 word preceding the core (namely “because” in (11)(h) and (11)(i) and “but” in (11)(e)). To illustrate this alignment, the same n-grams are written in columns in (12). Finally, all the words of the n-grams are compared position by position and the most specific common element is chosen to build the pattern. The decreasing order of specificity is: token, lemma, specific POS tag, generic POS tag, and wildcard. In PEP, the correspondence between specific and generic tags must be provided by the user. However, a correspondence for the Penn Treebank tagset is provided. In our example, the words appearing in position 1, 6-11 do not share any element since some of them are wildcards (represented by the asterisk); a wildcard will be used in the pattern for those positions. In position 2, 4 and 5, all the words share the same token, respectively “I”, “not” and “know”. In position 3, while all the n-grams are not made of the same token, they all share the same lemma “do”. The pattern built by PEP from that n-gram family is constituted by those elements, as written in (13). Words in bold and italics represent lemmas, both here and further in the chapter.

(11) *Examples of n-gram in the family “I do not know”.*

- (a) I did not know
- (b) I do not know
- (c) I did not know how
- (d) I did not know why
- (e) but I did not know how it would end
- (f) I did not know how it all started
- (g) I do know know why I like him so much
- (h) because I do not know why it is important
- (i) because I did not know if I would succeed

(12) *Examples of n-gram in the family “I do not know” aligned.*

	1	2	3	4	5	6	7	8	9, 10, 11
(a)	*	I	did	not	know	*	*	*	*
(b)	*	I	do	not	know	*	*	*	*
(c)	*	I	did	not	know	how	*	*	*
(d)	*	I	did	not	know	why	*	*	*
(e)	but	I	did	not	know	how	it	would	end
(f)	*	I	did	not	know	how	it	all	started
(g)	*	I	do	not	know	why	I	like	him so much
(h)	because	I	do	not	know	why	it	is	important
(i)	because	I	did	not	know	if	I	would	succeed

(13) * I **do** not know * * * * *

Once a pattern is created, the n-grams of the family are gathered into “sub-families” to build other, more precise patterns.

Let m be the number of lemmas in the common core of the parent family. Each sub-family will have a common lemma core of at least $m + 1$ lemmas; as a consequence, n-

grams of size m do not belong to any sub-family. The additional common lemma(s) could be either on the left or on the right of the parent core. Some n-grams that have lemmas both on the right and the left of the core could thus theoretically be part of two sub-families: one that would group all the n-grams that have the same lemma on the left of the parent core, and one that would group all the n-grams that have the same lemma on the right the parent core. To avoid repetition of n-grams, PEP puts each n-gram in only one sub-family; priority is given to the lemma on the right of the core. The n-grams of length greater than m of example (11) would be sorted as indicated in (14). We see that “because I do not know why this is important” and “because I do not know if I will succeed” are not grouped, even though they share the lemma “because” on the left of the core, because “because I do not know why this is important” share the lemma “why” on the right of the core with another n-gram, and priority is given to lemmas on the right of the core.

When a group contains more than one n-gram, PEP creates a new pattern with the n-grams in that group, as it is the case in (14)(a) and (14)(b). This continues recursively until no sub-group contain more than one n-gram. The pattern and sub-patterns created from the family given in (11) are written in (15) (lemmas and generic tags are written in bold italics, tokens and specific tags are written without font effect).

(14) *N-gram family “I do not know” grouped into sub-families.*

I do not know

(a) I do not know how

but I did not know how it would end

I do not know how it all started

(b) I do not know why

I do not know why I like him so much

because I do not know why it is important

(c) because I did not know if I would succeed

(15) *Pattern “I do not know” and its sub-patterns.*

I ***do*** not know

I ***do*** not know how

but I did not know how it would end

I do not know how it all started

I do not know why PP ***V***

I do not know why I like him so much

because I do not know why it is important

because I did not know if I would succeed

In summary, patterns and sub-patterns are condensed versions of several n-grams; they offer a single representation of several of them and, due to their hierarchical structure, they allow to observe easily different levels of precision.

1.1.4. Filtering patterns

So far, all n-grams are grouped into families and transformed into patterns. Those patterns are MWU candidates and they must be filtered to keep only the most relevant MWUs according to the criteria set by the user. Currently, PEP implements filters based on the frequency, the dispersion, the words composing the patterns and the properties of the pattern. Table 4 gives an overview of the current exhaustive list of filters implemented in PEP. Each of these filters is explained in this section.

Table 4: Filters implemented in PEP

Frequency	– Minimum frequency of main patterns (patterns that have no parent) – Minimum frequency of sub-patterns – Sub-pattern frequency relative to the parent pattern
Dispersion	– Minimum dispersion (for main and sub-patterns) – Minimum/maximum range – List of the corpora in which the patterns must appear
Words	– Lists of stop-words – Lists of words that must be included in the patterns
Pattern properties	– Minimum/maximum number of variants

First, there are **minimum raw frequency thresholds**. Frequency is one of the most widely used filters. In PEP, two different thresholds can be set: one for main patterns (i.e. patterns that are at the top of the hierarchy, that have no parent) and one for sub-patterns. If the same value is given to both thresholds, the resulting MWUs will only include main patterns; the results will be “flat”, all patterns will be at the top of the hierarchy. If a main pattern corresponds to the user’s setting and is displayed, there is a chance that its sub-patterns, even with a lower frequency, are also relevant to the user’s objectives since they are a more precise version of their parent pattern. On the other hand, low frequency main patterns are most of the time not phraseological because they represent only few low frequency n-grams. For that reason, it is often meaningful to set a lower frequency threshold for sub-patterns than for main patterns. The default frequency threshold for main patterns is set at 2, to remove single occurrences of n-grams and keep all others, and it is set at 1 for sub-patterns, so that no sub-pattern is filtered.

Better than a raw frequency threshold, PEP implements for the sub-patterns a **frequency threshold relative** to the frequency of their most direct parent. This threshold is

expressed as a proportion between 0 and 1. For example, if this threshold is set at 0.5, the sub-patterns must reach at least half of the frequency of their most direct parent to be displayed. In that case, if a pattern has 20 occurrences, only its sub-patterns of at least 10 occurrences will be displayed. If, among those sub-patterns, one has a frequency of 11, it will be displayed, and all its sub-patterns that have at least 6 occurrences will be displayed as well. This threshold is more powerful than the raw frequency threshold for sub-patterns, because it adapts itself to each level in the pattern hierarchy, and to each number of occurrences of the parent patterns. After trying several thresholds, one that seems to be suitable to identify sub-patterns that are meaningful is 0.1. Under that threshold, most sub-patterns do not seem to be phraseological. The default value of this threshold is set at 0 (no filter) so that, when using PEP for the first time, a user sees all the sub-patterns used to build it.

PEP does not currently implement frequency thresholds relative to the size of the corpus.

Next, several **dispersion** criteria are implemented. As pointed out by Gries (2008), dispersion through the corpus is an important element to identify and describe MWUs. To take it into account, PEP can process several files at the same time; each file is considered to be a sub-corpus. The user can set a maximum value of uneven dispersion that a pattern must have to be displayed. This value corresponds to Gries' DP (Gries 2009) presented in the chapter State of the Art, section Error: Reference source not found. Next to a dispersion value, the user can use a maximum or minimum range. The minimum range is useful to see only patterns that appear in several sub-corpora; those patterns will be more representative of the whole corpus, and not of just one of its parts; the maximum range, when set at 1, is useful to see only patterns that are specific to one sub-corpus. In order to allow for more precision than simple range, PEP also offers the possibility to specify in exactly which sub-corpus or sub-corpora the patterns must appear. In that case, the pattern should appear only in the specified sub-corpora, and not be included in any other. If we assume that each sub-corpus represents a different variety of language, this feature is very useful to compare language varieties and to analyse patterns that are variant-specific, i.e. that appear in one sub-corpus and are absent from the other sub-corpora. By default, no dispersion criteria is used to filter the patterns since PEP could work with a corpus that contains no sub-corpora, where dispersion values would not mean anything.

Then, several lists of **stop-words** and **"must-include" words** are implemented. For stop-words, following (Merkel & Andersson 2000), PEP implements different lists for words that should not appear in the first, the last or any position in a pattern. Furthermore, since PEP works on a lemmatised and tagged corpus, it is possible to use tokens, lemmas or specific tags as stop-words. This makes a total of $3 * 3 = 9$ lists of stop-words. In addition to the stop-words, the must-include words are words that must

appear in a pattern for it to be displayed. If several must-include words are specified, all of them must appear in the pattern. Once again, those words can be tokens, lemmas or specific tags. In all cases, the words can either be character strings or regular expressions (regex). The regex syntax used is the one defined by the python package `re`⁵. Note that regex should match the whole words and not just some part of it. Thanks to the power of regex, it is possible to negate a stop-word to transform it in a must-include word. This can be useful if a user wants a must-include word in initial or last position of the patterns, since for must-include words, no choice in the position is implemented as such in PEP. For example, the regex “[^I]” set as a stop-word (token) in first position of the n-grams will match (and thus discard) all patterns that do not start with the token “I”. Regex also allow to specify several must-include words among which only one needs to be in the pattern for it to be displayed. This is the case, for example, of the regex “I|you”, where the patterns must include either the token “I” or the token “you”. By default, punctuation marks and some symbols are set as stop-words. Since not all users might be familiar with regular expressions, by default, stop-words and must-include words are considered to be character strings. Note that the symbol “*” is always a stop-word (token) in PEP because it is the symbol used as a wildcard and having it as a token in the n-grams might produce errors the program. If this symbol is in a corpus and needs to be kept, it must be replaced by another symbol before launching PEP.

Finally, the user can set a minimum and/or maximum **number of variants** that a pattern must have to be displayed. This is useful to focus on MWUs that are more or less fixed. By default, there is no maximum nor minimum number of variants set.

The default values were set to fit two main objectives. The first is to help a user that would use PEP for the first time to understand intuitively how patterns are built. For that reason, no sub-pattern is filtered, so that all the sub-patterns used to build a parent pattern are displayed. The downside of this is that many sub-patterns will be displayed, most of which will not interest the user, but he or she will be able to filter them easily after understanding what a pattern is. The second objective is to filter patterns that would most of the time not be of interest to the user. This aims at reducing the number of results and at speeding up the extraction time to not discourage the user. This includes all patterns that contain a punctuation mark and main patterns with a frequency of 1. The default parameters can be changed by the user.

1.1.5. *Displaying patterns*

After filtering the MWU candidates, they can finally be printed either on the screen or in a file.

5 <https://docs.python.org/3.8/library/re.html>

PEP offers two ways of sorting the patterns: either by decreasing frequency or by increasing value of DP, representing an uneven dispersion. Both patterns and sub-patterns are sorted according to the chosen method.

The lemmas and generic tags are written between asterisks, which is the way to write italics in the markdown format. Each word of the patterns is written separated by a white space. On the right of the patterns, we find their total frequency (an integer), their DP value (dispersion, a number with three decimals), and their frequency in each sub-corpus (written between brackets, each value separated by a comma); all of these pieces of information are separated by a tab character.

All sub-patterns are written with one more indentation level (tab character) than their direct parent. On the left of the main patterns (the patterns with no parent) is written their rank.

An example of how a pattern and its sub-patterns are displayed by PEP is to be found in Text 1.

Text 1: *Example of a pattern and its sub-patterns displayed by PEP.*

1	I	*do*	not	know	6	0.000	[3, 3]									
	I	did	not	know	5	0.100	[3, 2]									
	I	did	not	know	WRB	5	0.100 [3, 2]									
		I	did	not	know	why	3 0.167 [1, 2]									
			Because	I	did	not	know	why	this	was	important	2	0.500 [0, 2]			
				I	did	not	know	why	I	liked	him	1	0.500 [1, 0]			
					I	did	not	know	how	2	0.500 [2, 0]					
						Because	I	did	not	know	how	this	all	started	1	0.500 [1, 0]
						But	I	did	not	know	how	it	would	end	1	0.500 [1, 0]
							Because	I	did	not	know	3	0.167 [1, 2]			
							I	do	not	know	if	I	will	succeed	1	0.500 [0, 1]

Thanks to the differences in indentation between parent and sub-patterns, it is possible with some text editors like Sublime Text or Notepad++ to hide sub-patterns of a given level in order to see only the patterns that interest the user. In that way, it is possible to explore the data interactively, focusing on a specific level of detail at a time. To illustrate that possibility, the pattern from Text 1 is shown in Sublime Text with some sub-patterns hidden in Figure 4; more precisely, the lines with the numbers 5-6 and 9-10 are hidden, they can be displayed back by clicking on the arrow on the left of the line that precedes them.

Figure 4: Extract of PEP's output in Sublime Text with some sub-patterns hidden

```

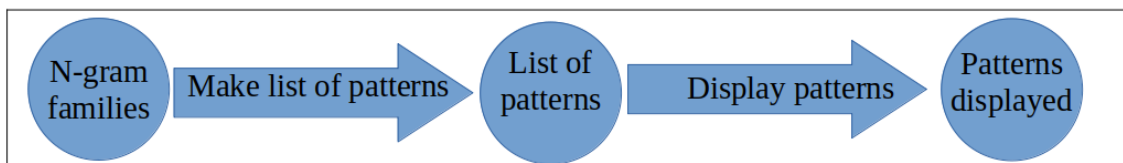
1 ▼ 1 I *do* not know 6 0.000 [3, 3]
2   I did not know 5 0.100 [3, 2]
3 ▼   I did not know WRB 5 0.100 [3, 2]
4 ►   I did not know why 3 0.167 [1, 2] ...
7
8 ►   I did not know how 2 0.500 [2, 0] ...
11
12   Because I did not know 3 0.167 [1, 2]
13   I do not know if I will succeed 1 0.500 [0, 1]

```

1.2 Extraction pipeline

PEP is programmed using a data flow structure managed by the python package `luigi`. This programming method divides the program into tasks, the output of some tasks being the input of others. This works in a similar way as an assembly line. This is illustrated in Figure 5: when a task, e.g. displaying the relevant patterns on the screen, needs to be done, `luigi` checks if all the required data, e.g. the whole list of patterns, is available. If it is the case, the data is used to perform the task, otherwise the previous task, e.g. making the list of patterns, is launched to produce the necessary data. The different tasks implemented in PEP are described in detail below; what was presented in the previous section (1.1) roughly corresponds to that.

Figure 5: Example of a data flow structure where tasks are represented by arrows and data by circles.



In the case of PEP, data flow programming has several advantages. First, when several extractions are made on the same corpus with different parameters, only the tasks depending on the modified parameters need to be rerun; the data from the previous tasks are still available and they need not be executed again. Second, the output of each task is stored on the computer disk and not the RAM; although this is slower, it avoids overloading the RAM with too much data. Third, by using the daemon scheduler provided by `luigi`⁶, it is possible to run several instances of the program at the same time; this allows for several independent tasks to be run at the same time and is thus faster than executing them one after the other. If this feature is used, `luigi` also offers a

⁶ This feature is only available on Unix systems (Linux/MacOS).

graphical visualization of the progress of the extraction pipeline in the form of a web page.

1.2.1. *Tagger*

The first task executed by the program is tagging the corpus. To do so, TreeTagger (Schmid 1995) is used, with the language specified by the user. There are several reasons why TreeTagger was chosen for this task. First, it performs both a lemmatisation and a part of speech tagging, which were both needed to fit the objectives of PEP. Second, TreeTagger offers the possibility to train new models, which broadens the possible customization of the extraction process with the program, which was also one of the objectives of PEP. Third, it works on all operating systems which is required to make the program available to users of all platforms.

1.2.2. *Xngrams & CreateNgramsBin*

From the output of the task **tagger**, n-grams of all the lengths specified by the user are extracted. The n-grams take into account the tokens, lemmas and tags provided by TreeTagger. As a consequence, if the same sequence of tokens has been tagged differently, they will be counted separately. All n-grams are first stored as a string (in a text file) and are then converted into a binary format which is more easily manipulable by the computer. Keeping the text file as well as the binary format has the advantage that an advanced user can read the text file to understand better from what n-grams PEP built the patterns of a corpus, and this file could even be re-used for purposes beyond PEP.

In the text file, the n-grams follow the format given in (16)⁷ and illustrated in (17) for the token n-gram “I did not know”.

(16) token1 tag1 lemma1||token2 tag2 lemma2|| ... ||tokenN
 tagN lemmaN freq

(17) I PP I||did VBD do||not RB not||know
 VB know 4

1.2.3. *GroupNgrams*

Once the n-grams are extracted from each sub-corpus for each length requested by the user, the n-grams from each corpus are grouped by length (one file per length). Even though the n-grams from the different sub-corpora are grouped, PEP keeps track of their frequency in each sub-corpus.

⁷ The blanks between each character string is a tab character.

1.2.4. *FilterNgrams & RemoveEmbeddedNgrams*

Next, the n-grams are filtered to keep only those that match the lists of the stop-words and must-include words specified by the user.

Once the n-grams not corresponding to the lists of words are filtered, the remaining ones are compared to remove any nested n-grams. Only nested n-grams that have exactly the same frequency as their (only) super-n-gram are discarded by PEP. All the others are kept for the remaining processing.

It is important that non relevant nested n-grams are discarded *after* filtering them out according to the stop-words and must-include words defined by the user. If those two steps were inverted, a relevant small n-gram might be discarded. To illustrate that, let's elaborate on what would happen if those steps were inverted. Take the example of a corpus containing the phrase "this is the end of the world" and a user who wants to extract from it patterns from 3 to 4 words. The user sets the stop-word "this". From that phrase, PEP would extract (among others) the n-grams "this is the end" and "is the end" (supposedly with the same frequency). If nested n-grams were discarded before considering the list of stop-words, "is the end" would be discarded because it is a non-relevant nested n-gram and "this is the end" would be discarded because it contains one of the stop-words specified by the user. As we see, in that case, nothing is left for further processing. On the contrary, since PEP first considers the list of stop-words, "this is the end" is discarded first and, as a consequence, "is the end" is no longer a nested n-gram; it is thus kept for further processing.

1.2.5. *GroupEmdeddedTags & BuildPatterns*

All the n-grams remaining at this step will be used to build patterns. Accordingly, they need to be grouped into families. For more efficient computing, this is done in two steps: First all the n-grams sharing a common core of generic tags are grouped. That core must be at least as long as the shortest pattern length asked by the user. Second, among each generic tag group, the n-grams sharing a common core of lemmas are grouped. Once again, that common core must be at least as long as the smallest pattern length asked by the user. Those n-grams correspond to a family and will be used to build patterns.

If an n-gram matches several, common cores (both of generic tags and of lemmas), it will be added to each group. This can be the case when an n-gram is longer than the smallest length requested by the user. For example, if a user requests patterns of length between 3 and 5, the n-gram "I do not know" could match both the lemma cores "I do not" and "do not know", and it would be added to both groups. As a consequence, while all main patterns appear only once, sub-patterns can appear under several main patterns.

Then, all the n-grams of a family are compared to each other. If two n-grams (e.g. (18) (a) and (18)(b)) have the maximum length requested by the user, say n (e.g. 4), if they have the same frequency in each sub-corpus and if they are made of the same words, except for the first word of the one (e.g. “I” in (a)) and the last word of the other (“week” in (b)), it is very probable that those two n-grams were extracted from the same sentence. To avoid counting them two times, they are merged into a single n-gram of length $n + 1$ (as in (c)). It might happen that two n-grams are merged even though they do not come from the same sentence(s) in the original corpus, but this is quite rare as long as the user requests patterns of at least three words. It is not recommended to extract patterns of less than three words. Since step greatly improves the accuracy of the results by not counting two times words that come from a single sentence, this step is currently always executed by PEP. Improvements will be made in future versions to avoid merging two n-grams that were not extracted from the same sentences.

(18) *Example of n-grams merged by PEP.*

- (a) I am coming next (frequency = 1)
- (b) am coming next week (frequency = 1)
- (c) => I am coming next week (frequency = 1)

Patterns are then built with the n-grams of the families by comparing all the words in each position of the n-grams and by selecting the most precise common element to constitute the pattern, as described more fully in section 1.1.3.

1.2.6. *DisplayPatterns*

Finally, the patterns are filtered based on their frequency, dispersion and number of variants to match the user’s specifications before being displayed in a hierarchical structure.

After displaying all the patterns, a numerical summary of the output is displayed. As showed in Text 2, several numbers are written, corresponding to four different categories: the number of pattern types displayed, the number of pattern tokens, the number of variants (types) that the patterns have, and the same for the variant tokens. The different numbers of each sequence, except for the first one, represent the frequency of the corresponding pattern length: the second number of the sequence corresponds to patterns of length 1, the third number, the patterns of length 2, etc. The first number of each sequence is the sum of all the other numbers. That summary is made, first for all corpora together, then for each corpus separately. In the example provided in Text 2, there is only one sub-corpus; the information under “all corpora” and “Corpus_example” is thus the same. Note that “Corpus_example” corresponds to the name of the file that representing that sub-corpus.

Text 2: Example of statistics displayed by PEP after displaying the patterns

```
All corpora:
Types:
7, 0, 0, 0, 0, 6, 1, 0, 0, 0

Tokens:
59, 0, 0, 0, 0, 53, 6, 0, 0, 0

Number of variants (types):
6, 0, 0, 0, 0, 6, 0, 0, 0, 0

Number of variants (tokens):
37, 0, 0, 0, 0, 37, 0, 0, 0, 0

Corpus_example:
Types:
7, 0, 0, 0, 0, 6, 1, 0, 0, 0

Tokens:
59, 0, 0, 0, 0, 53, 6, 0, 0, 0

Number of variants (types):
6, 0, 0, 0, 0, 6, 0, 0, 0, 0

Number of variants (tokens):
37, 0, 0, 0, 0, 37, 0, 0, 0, 0
```

Together with that numerical summary, the user can request a csv output format along with the standard output format described previously. The csv output contains only the main patterns, in the same order as in the standard output, along with their total frequency, their number of variants and then their frequency in each sub-corpus. This allows to have an easy access to the full distribution of frequencies and number of variants of the main patterns in order to perform statistical analyses. Unfortunately, there is currently no way to include such detailed information about the sub-patterns in the csv output format.

1.3 Short examples of PEP queries and results

To illustrate how the different parameters of PEP work and influence the results, I will launch different queries on the corpus sample in Text 3 and briefly discuss the results.

Text 3: Example corpus sample with two sub-corpora represented by the columns

But I did not know how it would end. Because I did not know how this all started. I did not know why I liked him.	Because I did not know why this was important. Because I did not know why this was important. I do not know if I will succeed.
---	--

For the first query, I requested patterns of length 3 to 9 (i.e. the length of the longest sentence), with only punctuation marks as stop-words (default parameters). I also used the default language parameters, that is “english” as TreeTagger language and the simplified tagset for the Penn Treebank tags provided with PEP. The resulting patterns are transcribed in Text 4.

Text 4: Query: PEP -i /path/to/corpus/ -m 3 -n 9

```

1 I *do* not know 6      0.000 [3, 3]
  I did not know   5      0.100 [3, 2]
  I did not know WRB 5      0.100 [3, 2]
    I did not know why 3      0.167 [1, 2]
      Because I did not know why this was important 2      0.500 [0, 2]
        I did not know why I liked him 1      0.500 [1, 0]

    I did not know how 2      0.500 [2, 0]
      Because I did not know how this all started 1      0.500 [1, 0]
        But I did not know how it would end 1      0.500 [1, 0]

  Because I did not know 3      0.167 [1, 2]
  I do not know if I will succeed 1      0.500 [0, 1]
```

Only one main pattern was created since all the n-grams extracted and kept by PEP share the lemma n-gram “i do not know”. Among all the n-grams that match this lemma-gram, several were followed by the specific tag “WRB” (wh-abverb) and were grouped to form the sub-pattern “I did not know WRB”. No other group could be formed at this level, because the remaining n-grams do not share any more consecutive tags than those in the parent common core. That sub-pattern itself has two variants: one in which the common lemma core is followed by “why” and one where it is followed by “how”. The patterns are sorted in decreasing order of frequency, since this is the default sorting method, but in this case, it happened to also correspond to the dispersion sorting.

If we do not want to find the token “know” in our results, we can add it as a stop word. In this case, it is a token stop word, that should not appear in any position of the n-gram. The results are transcribed in Text 5. This time, all the n-grams could not be grouped under the lemma n-gram “I do not know” since the word “know” cannot be used. Several n-grams could still be grouped under the lemma n-gram “I do not” and the remaining n-grams are displayed alone.

Text 5: Query: PEP -i /path/to/corpus/ --sw-mid-tk know . -n 9 -F 1

1	I *do* not	6	0.000	[3, 3]
	I did not	5	0.100	[3, 2]
	Because I did not	3	0.167	[1, 2]
	But I did not	1	0.500	[1, 0]
	I do not	1	0.500	[0, 1]
2	why this was important	2	0.500	[0, 2]
3	how this all started	1	0.500	[1, 0]
4	why I liked him	1	0.500	[1, 0]
5	how it would end	1	0.500	[1, 0]
6	if I will succeed	1	0.500	[0, 1]

In those results, all the n-grams extracted are displayed, and it could be argued that many of them are not phraseological. To restrict the results, it is possible to apply some filters. For example, we can assume that all patterns and sub-patterns that appear only one time are not frequent enough to be phraseological. Text 6 transcribes the results with those two filters. This time, there are only two main patterns: “I *do* not” and “why this was important”, which appear in the corpus six and two times respectively. “I *do* not” has two variants that have a frequency superior to one. Since the corpus sample is very small, those patterns are all quite representative its content.

Text 6: Query: PEP -i /path/to/corpus/ -n 9 --sw-mid-tk know . -E 2 -F 2

1	I *do* not	6	0.000	[3, 3]
	I did not	5	0.100	[3, 2]
	Because I did not	3	0.167	[1, 2]
2	why this was important	2	0.500	[0, 2]

Let’s come back to our initial results, where only punctuation was used a stop-words (default). If we want to see patterns that are used homogeneously in all parts of our corpus, we can set a maximum DP value (uneven dispersion), for example at 0.2. In that case, the results correspond to what is written in Text 7. As in Text 6, all the results displayed here seem to be quite representative of the recurrent expressions used in our corpus.

Text 7: Query: PEP -i /path/to/corpus -n 9 -D 0.2

1	I *do* not know	6	0.000	[3, 3]
	I did not know	5	0.100	[3, 2]
	I did not know WRB	5	0.100	[3, 2]
	I did not know why	3	0.167	[1, 2]
	Because I did not know	3	0.167	[1, 2]

2 Evaluation

In automatic phraseology extraction, when the kind of phraseological units that should be extracted are well defined, such as named-entities or phrasal verbs, the best way to evaluate the automatic extraction is to measure the precision. In the case of automatic MWU extraction, the precision represents the proportion of MWUs correctly identified among all the extracted MWUs, and the recall measures the proportion MWUs identified among all the candidates that are true MWUs. Measuring precision and recall is a very common way to evaluate NLP tasks, and it is described specifically for phraseology extraction in Evert & Krenn (2005). However, in the case of PEP, the focus is not on a specific kind of phraseological expression, but rather on a more general phraseological property: variation. To evaluate best how PEP can help study this notion of variation, I chose to revisit a study that uses this notion in its analysis, namely the study by Götz & Schilk (2011) on the comparison of English as a native language (ENL), English as a second language (ESL) and English as a foreign language (EFL). A summary of their study is written below.

This evaluation of PEP has the objectives, by replicating that study, to illustrate the possibilities and limits of PEP, to illustrate the differences in the results when the focus of the analysis is on patterns instead of 3-grams, and to demonstrate that PEP works also on not standard data (spoken language of ESL- and EFL-speakers).

Summary: Götz, Sandra & Schilk, Marco. 2011. *Formulaic sequences in spoken ENL, ESL and EFL: Focus on British English, Indian English and learner English of advanced German learners*.

The objective of the study is to compare frequent multi-word expressions (MWE) in ENL, ESL and EFL.

The introduction points out the differences in the communities and speakers of ESL and EFL. First, the target norm of ESL and EFL-speakers is not the same: while EFL speakers aim at producing a language similar to the one used by native speakers, ESL speakers use a variant of English that is institutionalized and therefore aim at speaking

that institutionalized form. Second, the language acquisition process also differs. In EFL-communities, the linguistic input is often limited to schools whereas for ESL-communities it is much more varied.

The authors formulate the following hypotheses:

- Due to the differences in the language acquisition process, MWE used in ESL and EFL will differ from those used in ENL.
- Because of the functional differences in the use of ESL and EFL, there will be differences in the range of communicative functions of MWE between ENL & ESL speakers.

The authors chose to use spoken data, first, because it is more suitable to see the differences in the functional range of usage and, then, because variant-specific MWE are more likely to be found in spoken language, which is more spontaneous than written language and is free from written conventions. Since no corpus provides at the same time speech in ENL, ESL and EFL, the authors decided to use a combination of comparable pairs of sub-corpora: the India part of the International Corpus of English (ICE) as a representative of ESL is compared to the Great Britain part of the same corpus and the German part of the Louvain International Database of Spoken English Interlanguage (LINDSEI) as a representative of EFL is compared to its native counterpart, the Louvain Corpus of Native English Conversation (LOCNEC). Only the first 86k words of each ICE corpus were used so that all four sub-corpora would have approximately the same size.

They then performed four steps to analyse the data: First they extracted all the recurrent 3-grams with a minimum frequency of 5 with WordSmith Tools 4. Second, using a Perl script, they identified what they call a “common nucleus” that is a list of all the 3-grams present in the 4 corpora. Third, they categorized manually the 3-grams into different discourse functions. Fourth, they analysed the variant-specific 3-grams according to corpus family (ICE-GB vs ICE-India, LINDSEI-GE vs. LOCNEC).

Their quantitative analysis of all recurrent 3-grams presented no significant difference in both the number of types and tokens of 3-grams between the two ICE corpora. However, the authors found significantly fewer types and tokens of 3-grams in the LINDSEI-GE compared to the LOCNEC. This might show that German learners have less access to formulaic language than ESL- and ENL-speakers. 3-grams are much more frequent in the LINDSEI-GE and LOCNEC than in the ICE corpora, but this might be due to differences in the corpus design: in the former corpora, speakers talk about very

similar topics and are thus more likely to use the same 3-grams, whereas the latter corpora compile data from broadcast discussions, interviews and unscripted speech, which cover more varied topics. When the same topic is broached by more speakers, there is a higher chance that the 5 occurrences threshold is reached, leading to more 3-grams retained for the analysis. The analysis of the common nucleus of 3-grams revealed that those n-grams are significantly more frequent in the LINDSEI-GE and the ICE-India than in their native counterpart. This shows that learners rely more strongly on the 3-gram nucleus than native speakers, and they show less variation as well as less variability in the use of formulaic language at the level of 3-grams. Those n-grams are more frequent in the LINDSEI-GE and in the LOCNEC, but this might once again be due to differences in the corpus design. As for the discourse functions fulfilled by the nuclear 3-grams, the authors observe that topic-oriented 3-grams are equally distributed across the ICE-corpora (c. 40%), but that there are more frequent in the LOCNEC (38%) than in the LINDSEI-GE (27%). They find almost no text-organizational nuclear 3-grams. They advance two hypotheses to explain that phenomenon: either text-organizational 3-grams are extremely variant -specific (and thus absent from the nucleus) or they are not internalized in one of the variants. They also note the absence of performance phenomena, which is probably due to differences in transcription conventions between the corpora. Finally, variant-specific 3-grams are more frequent (both types and tokens) in the LOCNEC and in the LINDSEI-GE than in the ICE corpora. This could be due to differences in the corpus design. There are significantly more type and token occurrences of variant-specific 3-grams in the ICE-India than in the ICE-GB however less variability could have been expected. There are significantly more variant-specific 3-gram in the LOCNEC than in the LINDSEI-GE, which could be a hint that EFL-speakers have less access to formulaic sequences than ENL-speakers. As for the categorical analysis, the authors found a stronger tendency towards using topic-oriented 3-grams for ESL and EFL, whereas there was a higher proportion of text-organisational 3-grams in both ENL-corpora. EFL uses the most performance phenomena (28%), followed by ENL (16% and 10%), ENL uses the least (4%).

To revisit this study, I first prepared the corpora in a similar way as Götz and Schilk did (section 2.1), then I replicated their quantitative analysis using PEP instead of WordSmith Tools (section 2.2), and finally, I conducted a qualitative analysis on the corpora (section 2.3). Please note that the aim of revisiting this study is not per se to compare the results obtained, but rather to illustrate how PEP can be used, and how examining patterns (as defined in PEP) instead of 3-grams can highlight other phenomena.

2.1 Preparing the data

To revisit the study, I worked on the same four corpora as the authors. The LINDSEI-GE and the LOCNEC contain speech from interviewers and interviewees. No information is provided about the interviewers, who might or might not be native speakers. Götz and Schilk did not make a difference between the utterances produced by the participants and those produced by the interviewers, which could introduce some bias in the data. To avoid that, I used only the utterances made by the participants. In that way, I ensured that only data from German native speakers were used in my analyses of the LINDSEI-GE, and, to keep the corpora comparable, I did the same for the LOCNEC. After that, I found that the smaller corpus I had was the LOCNEC_{participants} with 135,193 words. This is more than the size reported by Götz and Schilk, but they did not specify how they counted the words; here the words were counted by PEP, which counts as words all tokens identified by TreeTagger. Similar to what the authors did, I reduced the size of the other corpora so that the four samples would have approximately the same size; in my case, I considered approximately the 135k first words of the LOCNEC_{participants}, the ICE-GB and the ICE-India.

2.2 Quantitative analysis

In their paper, Götz and Schilk used log-likelihood tests to test the significance of the differences in type and token frequencies of the 3-grams they extracted. I used the same test as them, with a significance threshold set at 0.05.

In the study, the authors chose to analyse only token 3-grams probably because, as was discussed when WordSmith tools was presented (Chapter 2 section 1.3.2), this tool does not offer a good way to handle nested n-grams and 3-grams is a common length for that kind of study. Since PEP's focus is on nested n-grams, I extracted patterns from 3 to 5 words. Those are usually the n-gram lengths studied in computational phraseology, and it includes the length chosen by Götz and Schilk. As did the authors, I set a minimum frequency threshold to 5, both for the main patterns and the sub-patterns. Because non-standard varieties of English are studied, I did not want to limit the results to a particular kind of MWUs, and I adopted an exploratory approach. Therefore, I left the other parameters to their default values, which include punctuation marks as stop-words, as seems to be the case in the original study. This extraction was made on each corpus independently so that the frequency of the n-grams in one corpus would not influence the extraction of patterns from the other corpora, as was the case in the study.

At this step, the authors reported that there was no statistical difference in the number of types and tokens in the two ICE corpora. However, they found significantly fewer types and tokens in the LINDSEI-GE than in the LOCNEC. Concerning the LINDSEI-GE and the LOCNEC, as we can see in Table 5, I found the same results as the authors:

there were significantly more pattern types and tokens in the LOCNEC than in the LINDSEI. However, while the authors did not find any significant difference between the number of types and tokens of 3-grams in the two ICE corpora, I found significantly more occurrences of pattern tokens in the ICE-GB than in the ICE-India. However, while the ICE-GB did contain more pattern types, that difference is not significant. The authors also noted an “enormous difference in the frequencies of 3-grams between the corpus families: [...] there are roughly twice as many 3-grams in LINDSEI-GE/LOCNEC1–41 than there are in the two ICE86k-corpora” (2011: 88). They explained this difference with the corpus design: the topics of the talks in the LOCNEC and the LINDSEI-GE are much less varied than in the two ICE corpora; as a consequence, the threshold of 5 is supposedly reached faster in the LOCNEC/LINDSEI-GE than in the ICE corpora. However, here I did not find such a difference between the two corpus families, but only the LOCNEC contained almost twice as many patterns than the other corpora, which seems to contradict the author’s theory.

Table 5: Overview of all patterns in the corpora

	ICE-India (135k)	ICE-GB (135k)	LOCNEC (participants)	LINDSEI-GE (participants - 135k)
Corpus size	135,400	135,872	135,193	136,766
Number of pattern types	843	865	1,434	802
Number of pattern tokens	7,774	8,310	14,768	7,754
Types p-value	$p > 0.05$ ($G^2 = 0.21$)		$p < 0.05$ ($G^2 = 173.81$)	
Tokens p-value	$P < 0.05$ ($G^2 = 16.05$)		$p < 0.05$ ($G^2 = 2140.23$)	

Subsequently, the authors analysed more specifically the 3-grams that were common to all corpora. That’s what they called the “nucleus”. To identify that nucleus, they used a Perl script that compared the lists of 3-grams two by two, and kept those that appeared in both lists. With PEP, to extract a nucleus of patterns that are common to all corpora, I simply set the minimum range to be equal to 4, which means that only the patterns that appear in the four corpora were displayed. The first thing to note is the huge difference in the number of nucleus patterns extracted with PEP and the number of nucleus 3-grams found in the original study: 1586 against 54. This difference can easily be explained by the fact that a pattern groups several token n-grams that have the same lemma n-gram. This means that the threshold of 5 occurrences is more easily reachable by patterns than by token n-grams, like in the original study. The authors find that the nucleus n-grams appear much more frequently in the LOCNEC and LINDSEI-GE than

in the ICE corpora, which they explain once more with the differences in the corpus design. They also find that the n-grams of the nucleus are significantly more frequent in the LINDSEI-GE and the ICE-India than in their native counterpart. Götz and Schilk explain this by saying that, at the level of 3-grams, learners rely more strongly on the nucleus and show less variability in their use of formulaic language than native speakers. Here I found significant results that are opposite to the findings of the original study: as reported in Table 6, I found significantly fewer occurrences of the nucleus patterns in the ICE-India and in the LINDSEI-GE than in their native counterpart. If the learners rely more on the nucleus 3-grams than native speakers, as observed by Götz and Schilk, it seems that it is not true for patterns. Patterns allow for much more variability than n-grams, variability which might be harder for learners to master. To check this hypothesis, I compared the average number of sub-patterns that we find in each corpus. To do so, I used the csv output format of PEP and computed with R the average and standard deviation of the number of sub-patterns found in each corpus. After checking the normality of the data, I ran a two-tailed t-test on each corpus family. I found that patterns in the LINDSEI-GE have significantly fewer sub-patterns than the patterns in the LOCNEC ($t = -3.5524$, $df = 1969.8$, $p\text{-value} = 0.0003907$). I did not find a significant difference in the number of sub-patterns in the two ICE corpora ($t = 0.37177$, $df = 1700.8$, $p\text{-value} = 0.7101$). Those results seem to indicate that, indeed, EFL-speakers use patterns that have less sub-patterns – and therefore that they are more fixed – than ENL-speakers. However, the same cannot be confirmed for ESL-speakers.

Table 6: Overview of the nucleus patterns in the corpora

	ICE-India (135k)	ICE-GB (135k)	LOCNEC (participants)	LINDSEI-GE (participants - 135k)
Corpus size	135,400	135,872	135,193	136,766
Number of pattern tokens	6,465	7,861	10,826	7,626
Tokens p-value	$p < 0.05$ ($G^2 = 131.43$)		$p < 0.05$ ($G^2 = 595.40$)	

Finally, the authors compared the variant-specific n-grams, i.e. the n-grams that appear in only one of the four corpora. They found significantly more occurrences of types and tokens in the ICE-India than in the ICE-GB, although they were expecting the opposite results. In the other corpus family, they found significantly more types and tokens of 3-grams in the LOCNEC than in the LINDSEI-GE, which corroborated their hypothesis. I found the same results with the variant-specific patterns, as reported in Table 7.

Table 7: Overview of the corpus-specific patterns

	ICE-India (135k)	ICE-GB (135k)	LOCNEC (participants)	LINDSEI-GE (participants - 135k)
Corpus size	135,400	135,872	135,193	136,766
Number of pattern types	126	55	122	40
Number of pattern tokens	585	342	747	260
Types p-value	p < 0.05 ($G^2 = 28.86$)		p < 0.05 ($G^2 = 44.44$)	
Tokens p-value	P < 0.05 ($G^2 = 65.30$)		p < 0.05 ($G^2 = 251.35$)	

2.3 Qualitative analysis

In addition to the quantitative analysis, the authors performed a qualitative analysis. This was carried out by categorizing manually all the n-grams of the nucleus and all the n-grams specific to one of the studied varieties of English into the categories defined in Table 8. They did not define them further than by giving some examples, and those examples can be ambiguous. For example, they classify *as far as* as text-organisational. However, that 3-gram could be classified in different categories depending on the context in which it appears: when it is part of the longer phrase “as far as I’m concerned”, it is more speaker-oriented whereas, when it is part of the phrase “as far as possible”, it would rather be topic-oriented, and in the phrase “as far as X is concerned”, text-organisational. It is not clear if the authors looked at the context of the 3-grams to classify them.

Due to the lack of the lack of guidelines for the classification in Götz and Schilk’s article, I could not reproduce it strictly. Instead, I decided to define categories, inspired by theirs, in a way that could be automatically handled by PEP. The objective is to illustrate how the categorisation process, which is often completed manually, can be (at least partly) automatised using the stop-word and must-include word list features provided by PEP. This section first describes how the different pattern categories were defined, then followed by a qualitative analysis of the patterns, similar to the one conducted by Götz and Schilk is conducted, but without comparing the results with theirs since the categories are not strictly the same.

Table 8: Functional categories of 3-grams with examples from Götz & Schilk (2011: 87)

Functional category	Examples
Speaker-oriented 3-grams	I would think, I don't know, I thought about
Topic-oriented 3-grams	this is a, a lot of, the election commission
Text-organizational 3-grams	and of course, as far as
Audience-oriented 3-grams	and you can, you want to
Performance phenomena	and erm well, the the the, I it was
Others	of it so, it to all, so no I

2.3.1. Definition of pattern categories

The definition of the pattern categories was organised in three steps. First, some rules were inferred, based on the 3-gram examples given by Götz and Schilk, to know to which category a 3-gram (or pattern) belongs. Then, the literature on the classification of phraseological expressions was reviewed to compare the inferred rules to what is usually done in phraseology classification. Finally, the precision of the automatic classification performed by PEP was manually evaluated on a sample of the spoken part of the Corpus of Contemporary American English (COCA). Those three steps are elaborated below.

PEP implements several lists of stop-words and must-include words. That feature can be used to extract patterns that would correspond to a given category. Therefore, I decided to identify key-words for each category, based on the examples given by Götz and Schilk.

First, all the 3-grams in the speaker-oriented category contain the word “I”, this is the first key-word selected for that category. I decided to add the lemmas “me”, “we” and “us” because those are words that are also focused on the speakers⁸. In a similar way, all the 3-grams in the audience-oriented category contain the word “you”; this is the (only) key-word identified for that category.

For the topic-oriented n-grams, the authors reported many examples (see p. 94), most of which are sequences that are grammatically incomplete and do not bear a meaning that is really “topic-oriented” (e.g. “this is a”, “there was a”, “which is the”). I decided to refine that category to patterns that contain a noun, since nouns refer to a concrete

⁸ Setting must-include words as lemmas instead of tokens is a way to ignore capitalization.

reality and, since they are mentioned by the speaker, must somehow be related to the topic of the conversation.

The text-organizational category seems to be the hardest category to identify based only on the internal structure of the MWUs, without having access to the context in which the n-grams or patterns appear. In written language, punctuation might be a hint to identify them, but punctuation is not relevant in spoken language. In their study, Götz and Schilk identify only two n-grams as text-organizational, namely “and of course” and “as far as”. Because of the very small proportion of MWUs that the authors identified in that category, I decided to not take it into account for the qualitative analysis.

Götz and Schilk divided the performance phenomena in three categories: self-corrections, repetitions and hesitations. Self-corrections represent a very small proportion of the performance phenomena they identify – less than 5%. Like in “the plane er train”, hesitations are often made of words of the same part of speech that follow each other, sometimes separated by a hesitation marker. It is not possible to identify such sequences automatically with PEP because it does not enable the user to specify a sequence of POS that should be used in the patterns (at least in the current version). The same problem arises to identify repetitions. For that reason, I will not take into account all performance phenomena and focus on hesitations.

Table 9 gives an overview of the key-words selected so far.

Table 9: Temporary implementation of different pattern categories with PEP

Category	Definition
Speaker-oriented patterns	Must contain the lemmas “I”, “me”, “we”, or “us” and no hesitation key-word.
Topic-oriented	Must contain the generic tag “NN”, “NNS”, “NP”, “NPS”, and no key-word from another category.
Audience-oriented patterns	Must contain the lemma “you” and no hesitation nor speaker-oriented key-word.
Hesitations	Must contain the tokens “er”, “erm”, “mm”, “hmm”, “mhm”, “hm”, “uh”, “uhm” with or with a capital letter.
Other	Must contain none of the key-words from the other categories.

After selecting those key-words, I quickly reviewed the literature on the classification of phraseological expressions (PEs) – and more precisely of lexical bundles, as defined by Biber et al. (1999) – to refine the key-word selection. Two types of classifications can be distinguished: structural classifications, based on the grammatical structure of the PEs, and functional classifications, based on the pragmatic functions fulfilled by the PEs. The former kind of classification could be automatically implemented by defining sequences of POS tags that correspond to one category (examples of such classifications can be found in Biber et al. 1999; Hyland 2008; Hyland & Jiang 2018; Chen & Baker 2010; Cortes 2002). Unfortunately, this is not possible in the current version of PEP. As a consequence, I decided to focus on a functional classification, as did Götz and Schilk.

The vast majority of studies propose classifications for academic writing PEs, and none of them, except the one conducted by Götz & Schilk (2011), classify PEs of spoken language based on their functions and not on their internal structure. The classifications made for academic writing cannot be used as such for spoken language, but they need to be partly adapted.

The most widely used functional categories are those defined by Hyland (2008) and summarized in Table 10. The other classifications found in the literature are similar to that one in the main categories, and differ slightly in the sub-categories. For example, Chen & Backer (2010) distinguish as main categories *referential expressions*, *stance bundles* and *discourse organisers*, and Cortes (2004) identifies *referential bundles*, *text organizers* and *others bundles*. To refine my categories and their definitions, I decided to take a closer look at Hyland's classification because it is the most commonly used one. Based on his classification, I tried to see if my own classification was meaningful, and if the different key-words were well chosen.

Table 10: Functional categories for academic phraseological expressions by Hyland (2008)

Main category	Sub-category	Examples
Research-oriented	Location	<i>at the beginning of, at the same time, in the present study</i>
	Procedure	<i>the use of the, the purpose of the</i>
	Quantification	<i>the magnitude of the, one of the most</i>
	Description	<i>the structure of the, the size of the</i>
	Topic	<i>in the Hong Kong, the currency board system</i>
Text-oriented	Transition signals	<i>on the other hand, in addition to the</i>
	Resultative signals	<i>as a result of, it was found that</i>
	Structuring signals	<i>in the present study, as shown in fig.</i>
	Framing signals	<i>in the case of, with respect to the, on the basis of</i>
Participant-oriented	Stance features	<i>Are likely to be, may be due to</i>
	Engagement features	<i>it should be noted that, as can be seen</i>

Hyland’s research-oriented category contains the sub-category “topic”, which is one of the categories used by Götz and Schilk. However, they included in that category 3-grams such as “the first time”, “in front of” and “a little bit”, which correspond to other sub-categories of Hyland’s research-oriented main category (namely location and quantification). As a result, it seems that Götz and Schilk’s topic-oriented category corresponds to the whole research-oriented category defined by Hyland, and not to one specific category. Hence, I decided to use this whole category to refine the key-words chosen to identify automatically patterns belonging to the topic-oriented category. Except for “one of the most”, all the n-grams in that category include a noun; this comforts my choice to use that POS tag as a criterion to extract patterns that belong to that category.

The text-oriented category seems quite similar to what was intended by Götz and Schilk for their “text-organizational” category. The two sub-categories “resultative signals” and

“structuring signals” are clearly related to academic writing and do not seem relevant in speech. On the other hand, MWUs used in speech could belong to “transition signals” and “framing signals”, although probably not exactly the same MWUs as in academic writing. No token, lemma nor tag seems to be specific to this category, which makes them hard to identify with PEP. Since they were very marginal in Götz and Schilk’s analysis-- only 2 3-grams out of 54 –, I decided to not take that category into account.

The participant-oriented category includes the stance features, which are all PEs that reveals, in a certain way, the opinion of the writer, and the engagement features, that address the readers. Those categories correspond to what Götz and Schilk called “speaker-oriented” and “audience-oriented” for speech. While the use of first and second personal pronouns is avoided in academic writing, they are much used in spoken language, therefore, I decided to keep them as key-words for my extraction. However, based on Hyland’s examples, I decided to add the stance markers “may” and “might” to the list of key-words for the speaker-oriented category. No token, lemma nor tag seems common to the examples given by Hyland for the “engagement features” subcategory; I decided to keep my key-words for the audience-oriented category unchanged.

Table 11 summarises the final key-words list.

Table 11: Final implementation of different pattern categories with PEP

Category	Definition
Speaker-oriented patterns	Must contain the lemmas “I”, “me”, “we”, “us”, “may” or “might” and no hesitation key-word.
Topic-oriented	Must contain the generic tag “NN”, “NNS”, “NP”, “NPS”, and no key-word from another category.
Audience-oriented patterns	Must contain the lemma “you” and no hesitation nor speaker-oriented key-word.
Hesitations	Must contain the tokens “er”, “erm”, “mm”, “hmm”, “mhm”, “hm”, “uh”, “uhm” with or with a capital letter.
Other	Must contain none of the key-words from the other categories.

To evaluate the precision of that key-word based classification, I tested it with PEP on an extract of the spoken part of the Corpus of Contemporary American English (COCA) and manually checked the 100 first results. I chose to test the key-words on a corpus on which I am not conducting an analysis so that my choice of key-words would not be influenced by any of the four corpora used the analysis.

The patterns identified as speaker-oriented showed a precision of about 96%. Among the four problematic patterns identified, three are clear mistakes. First, “**let** me ask you” and “for joining us” (with “thank you for joining us” and “thanks for joining us” as sub-patterns) should be identified as audience-oriented and “joining us now” is most often used as text-organisational on TV shows (e.g. “And joining us now is Lucas...”, “She’s joining us now live from Columbia”). There is a fourth pattern that was identified as problematic, but its case is more arguable: “I think you can”. This pattern could be both considered as speaker- and audience-oriented. In the way I implemented the different categories, I decided to make the different categories mutually exclusive, and I gave the priority to the speaker-oriented category over the audience-oriented one. This decision is quite arbitrary, but it can easily be changed by adapting PEP’s settings.

For the audience-oriented category, I obtained a precision score of 100%. The real precision is most probably not exactly 100%, but it must be quite close to that. I found three different ways that are used to address the interlocutor(s): talking about hypothetical situations (e.g. “if you **be**”), asking direct questions (e.g. “**do** you know”) or addressing the interlocutor(s) directly (e.g. “you **be** going to”, “you **have** got to”). Among patterns that would be erroneously identified as audience-oriented by PEP, there could be repetitions (e.g. hypothetically “and you you”) that should be classified among the performance phenomena, but this cannot be detected by PEP in its current version.

The topic-oriented category showed the worst precision, with a score of 91%. Among the patterns wrongly identified as belonging to this category, there are “by the way”, “**in** order to” and “in a moment” that should be classified as text-organisational, “**thanks** very much” and “wait a minute” that belong to the audience-oriented category and “and of course”, which is a marker of stance and thus speaker-oriented (e.g. “...moving towards a regulated market economy , and of course , one of the first consequences is that rise of prices”). The three remaining patterns identified as problematic, namely “**at** this point”, “at that point” and “will be back”, can be either topic-oriented or text-organisational depending on their context of use. Table 12 gives examples from the COCA for those three patterns and each possible category.

Table 12: Examples of patterns that could belong to several functional categories

Pattern	Examples
<i>at</i> this point	Topic-oriented: “You’re so close at this point” Text-organisational: “So at this point, with credit now treated essentially as...”
at that point	Topic-oriented: “But I don't think that we're at that point with...” Text-organisational: “So , at that point , we decided to withdraw”
will be back	Topic-oriented: “President Obama will be back on more comfortable ground” Text-organisational: “We will be back shortly”

Finally, after trying the hesitation key-words on the COCA, I got no results because, in that corpus, hesitations are always transcribed between punctuation marks that I chose to filter out. Since the list of hesitation key-words include all hesitation markers found in the four corpora that I am analysing, I did not modify that list of key-words.

Overall, the precision of the automatic classification – i.e. what proportion of the identified patterns is correctly identified – is quite good. To know precisely how good the recall is – i.e. to know what proportion of all the patterns belonging to each category were identified –, it would be necessary to manually classify all the patterns belonging to the “other” category (that is, all the patterns that were not identified as belonging to any category). Since the objective of this classification is to demonstrate how PEP can be used to ease the process by reducing the manual work needed, it does not seem necessary to do such a precise recall analysis. More globally, there were 23% of the patterns that were not classified in one of the defined categories, and thus belong to the “other” category; as a consequence, the general recall of the automatic classification must be smaller than 23%.

2.3.2. Categorical analysis of the patterns in the corpora

Figure 6 gives the distribution of the nuclear patterns from the different corpora into the categories defined previously and Figure 7 the distribution of the variant-specific patterns. In all corpora, most patterns in the nucleus are speaker-oriented patterns. They are roughly as frequent in the LINDSEI-GE and in the LOCNEC, and are more frequent in the ICE-India than in the ICE-GB. Among the variant-specific patterns, it is the opposite: speaker-oriented patterns are more frequent in the ICE-GB than in the ICE-India, and they are still roughly equally distributed among the LINDSEI-GE and the

LOCNEC. This seems to indicate that ENL-speakers have more varied ways to involve themselves into the conversation than do the ESL-speakers who rely on more common patterns to do so.

In the nucleus, the topic-oriented patterns are equally distributed across all four corpora. That category is the one favoured in all corpora for the topic-specific patterns. This is probably due to the different topics broached in each corpus; this results in different words and, as a consequence, different patterns used in each corpus.

As pointed out by Götz and Schilk, the lack of performance phenomena (here hesitations) in the nucleus is due to the differences in transcriptions in the different corpora. More specifically, no hesitation patterns were extracted from the LINDSEI-GE because, in that corpus, all hesitations are written between parentheses, and all patterns containing parenthesis were filtered out to fit to the stop-words specified for the extraction. By looking at the variant-specific hesitations, we observe that they are more frequent in the ICE-GB than in the ICE-India, as was noted by Götz and Schilk.

A final interesting observation is that the ICE-India and the LINDSEI-GE contain more variant-specific patterns in the topic-specific and the “other” categories than their native counterpart. This could mean that the formulaic language specific to ESL- and EFL-speakers relies more strongly on the use of nouns.

Figure 6: Categorical analysis of the pattern nucleus across the corpora

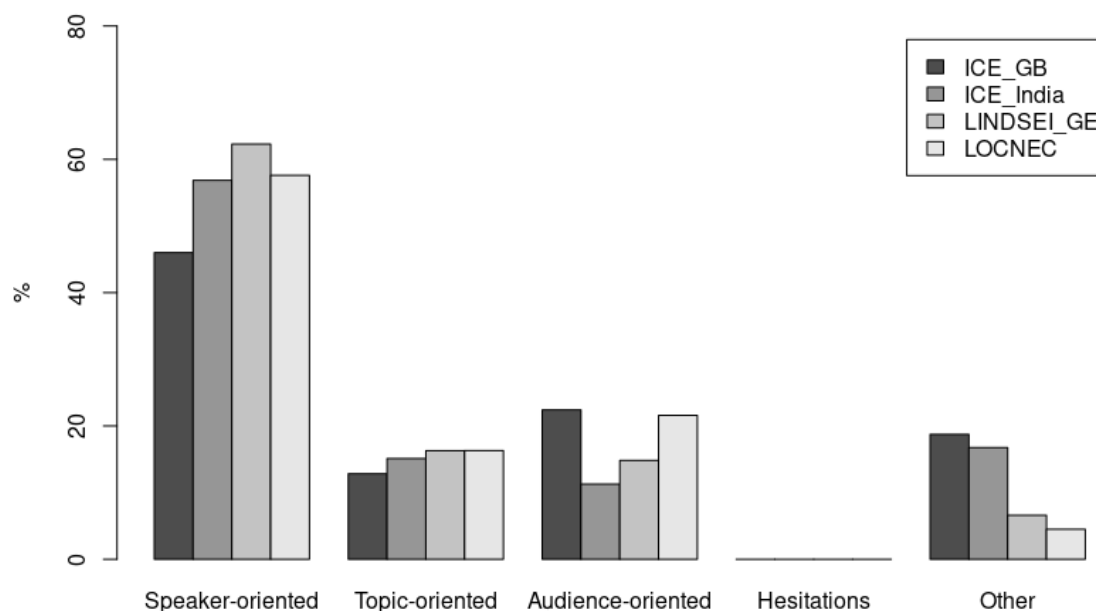
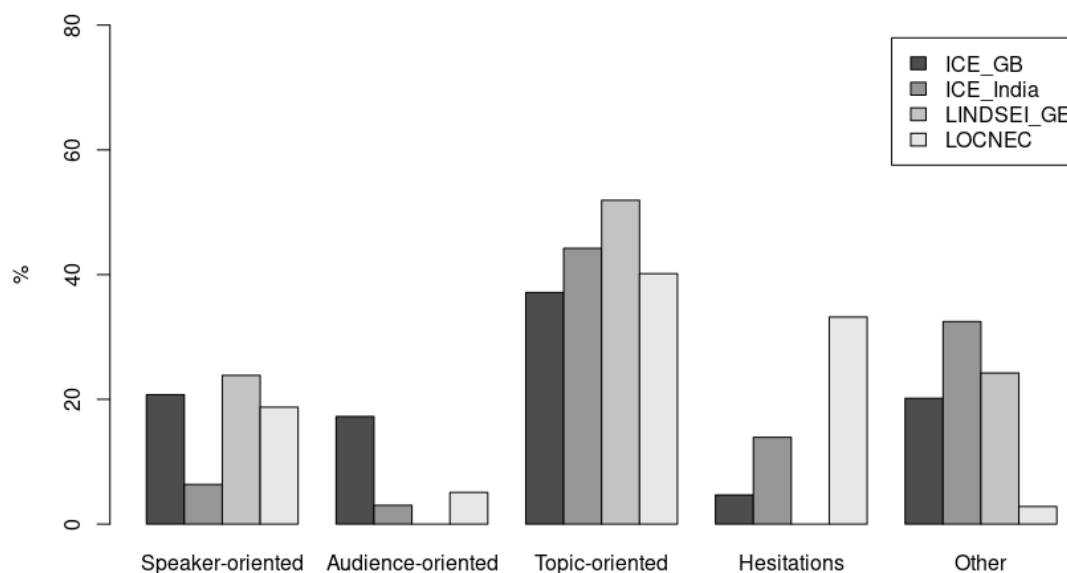


Figure 7: Categorical analysis of the variant-specific patterns across the corpora



2.4 Discussion of the results

By revisiting the study made by Götz & Schilk (2011) using PEP instead of WordSmith Tools, I illustrated how PEP can be used both for a quantitative and a qualitative analysis of MWUs. In the quantitative analysis, it was found that a different reality is highlighted when the focus is on patterns instead of 3-grams. In particular, PEP allows to know the degree of variation that the patterns show by including their sub-patterns in the analysis. With regard to the qualitative analysis, PEP makes it easy to search for patterns that contain specific key-words (tokens, lemmas or tags) while filtering out non-wanted words. It has been demonstrated that this feature can be used to sort patterns into functional categories with a high precision and a reasonable recall. This feature can still be improved by allowing the user to specify a sequence of words that the patterns should follow (e.g. determiner + adjective + noun). Moreover, this would allow to analyse structural categories next to functional ones. PEP has shown to easily allow to extract patterns that are specific to one corpus, or that are common to several corpora, and thus analyse variant-specific patterns or more general ones.

By using spoken data, it was demonstrated that PEP works well on less standard data. Even though TreeTagger commits many mistakes in the tagging procedure because of the spoken data, those mistakes do not seem to have a strong impact on the results, since the extraction of patterns does not only rely on tags, but also on lemmas and tokens.

3 Limitations & future work

PEP is still work in progress, and it needs to be improved in many ways.

The most important question that is yet to be answered is how to select the best parent pattern. In its current state, PEP simply groups n-grams into families, builds patterns and sub-patterns out of them, and prints the patterns that match the user's query. However, it may be the case, that a sub-pattern is more relevant than its parent pattern; in its current state, PEP completely ignores sub-patterns if their parent pattern does not match the user query. A possible way to identify the relevance of patterns and sub-patterns could be to use association measures. Another approach to be explored is using measures similar to the variant/p-frame ratio and the predictability that can be applied on p-frames (see Chapter 2, section 1.2.3). In the case of patterns, such measures could take into account the number of different variants that a pattern has, and how likely those variants are to be used. If a variant is not likely at all, it could be filtered out, and a pattern that has too many different variants could maybe be split into several different patterns, to avoid grouping together sub-patterns that are not strongly related to each other. Also, currently, if a pattern does not match the user's query, its children will not be displayed either all, even if some of them might correspond to the query.

A current limitation of PEP is that it is dependent on TreeTagger for the POS tagging, and TreeTagger is not always accurate, especially on less standard data, such as spoken data. To quantify the impact of TreeTagger mistakes on the results, we would need to run the same analysis, once tagged by TreeTagger, once tagged (semi-)manually and compare the results. Unfortunately, PEP does not currently offer to extract patterns from a pre-tagged corpus

As mentioned during PEP's evaluation, it is also not possible to extract specific sequences of tokens, lemmas or tags for the extraction to limit the results to, for example, noun phrases in the form of a determiner, followed by an adjective, followed by a noun.

Finally, the current text output, even if it can be explored interactively with some text editors, is not very user-friendly. Some work has still to be done to make the output lighter (currently, output files may be huge), maybe by limiting the information asked by the user, and to make it easier to explore interactively, maybe in an html format.

Since PEP is an open-source package, if a feature that is necessary for a study is missing, a user with some knowledge in python programming could implement that feature and share it with others to integrate it in future PEP versions.

Conclusion

The objectives of this thesis were to describe and evaluate the Phraseology Extraction Package (PEP), a program that I developed to automatically extract multi-word units (MWUs). I started by introducing the concept of phraseology by providing an overview of different definitions that can be given to word combinations, and of the many criteria that can be used to identify them. Then, I presented thoroughly the two main methodologies used to automatically extract MWUs, namely the cluster analysis, which focuses on recurrent continuous sequences of words, and the co-occurrence analysis, which uses the strength of association between groups of words to identify MWUs. In both cases, I showed that the different tools that are available to non-computational linguists to apply those methodologies in their study of phraseology do not take into account all the parameters that can be used to customize the extraction of MWUs. This limits the findings of the studies to what is permitted by the tools. To answer that problem, I have been developing a highly customizable program to automatically extract MWUs. That program, the Phraseology Extraction Package (PEP), was described and evaluated in the last chapter.

PEP has made two main contributions to the study of phraseology: 1) to provide a tool to extract and handle nested n-grams and 2) to integrate more variation and flexibility in the MWUs by including not only tokens, but also lemmas and POS tags in the MWUs. Because PEP's focus is not on a specific kind of MWUs, but rather on variation in phraseology, it was not possible to make a standard precision-recall evaluation. Instead, I revisited a study, originally made on 3-grams with WordSmith Tools, using PEP to illustrate the possibilities and limitations of the program. The evaluation showed that a different reality is highlighted when the study is made on 3-grams or on patterns output by PEP. Thanks to the many parameters that can be customized, it was shown that PEP facilitates the comparison of the patterns in different corpora as well as the categorisation of those patterns. Even though PEP could show good results in revisiting the study, there is much work to be done to improve it, in particular to improve the way patterns are built, and to make PEP more efficient to use.

Acknowledgments

Many people contributed, in their own way, to this thesis, and I would like to thank them all. First of all, I want to thank my supervisor, Dr Magali Paquot, for her constructive feedback that helped improve both the content and the form of the thesis. I also want to thank my colleague at the CECL, Bert Van de Poel, for testing PEP and giving many valuable ideas on how it could be improved. Thanks to my brother, Julien Castiaux, for his help and support with python programming. Finally, a huge thanks to Utku Dürüst and Yaşar Dürüst for proofreading the thesis.

References

- Aarts, Jan & Granger, Sylviane. 1998. Tag sequences in learner corpora: a key to interlanguage grammar and discourse. *Learner English on Computer*, 132–141. 1st edn. London and New York Addison Wesley Longman.
- Ackerley, Katherine. 2017. Effects of corpus-based instruction on phraseology in learner English. *Language Learning & Technology* 21(3). 195–216.
- Ackermann, Kirsten & Chen, Yu-Hua. 2013. Developing the Academic Collocation List (ACL) – A corpus-driven and expert-judged approach. *Journal of English for Academic Purposes* 12(4). 235–247.
- Adelman, James S. & Brown, Gordon D. A. & Quesada, José F. 2006. Contextual diversity, not word frequency, determines word-naming and lexical decision times. *Psychological Science* 17(9). 814–823.
- Altenberg, Bengt. 1998. On the phraseology of spoken English: The evidence of recurrent word-combinations. In Cowie, A. P. (ed.), *Phraseology. Theory, Analysis, and Applications*, 101–122. Oxford: Oxford University Press.
- Ambridge, Ben & Theakston, Anna L. & Lieven, Elena V. M. & Tomasello, Michael. 2006. The distributed learning effect for children’s acquisition of an abstract syntactic construction. *Cognitive Development* 21(2). 174–193.
- Anthony, Laurence. 2019. *AntConc*. Tokyo, Japan: Waseda University. (<https://www.laurenceanthony.net/software>) (Accessed August 12, 2020.)
- Arnon, Inbal & Snider, Neal. 2010. More than words: Frequency effects for multi-word phrases. *Journal of Memory and Language* 62. 67–82.
- Backus, Ad. 2003. Units in code switching: evidence for multimorphemic elements in the lexicon. *Linguistics* 41(1). 83–132.
- Banerjee, Satanjeev & Pedersen, Ted. 2003. The Design, Implementation, and Use of the Ngram Statistics Package. In Gelbukh, Alexander (ed.), *Computational Linguistics and Intelligent Text Processing* (Lecture Notes in Computer Science), 370–381. Berlin, Heidelberg: Springer.
- Bestgen, Yves. 2018. Evaluating the frequency threshold for selecting lexical bundles by means of an extension of the Fisher’s exact test. *Corpora* 13(2). 205–228.

- Biber, Douglas. 2009. A corpus-driven approach to formulaic language in english: Multi-word patterns in speech and writing. *International Journal of Corpus Linguistics* 14(3). 275–311.
- Biber, Douglas & Gray, Bethany. 2016. *Grammatical complexity in academic English: linguistic change in writing* (Studies in English Language). Cambridge: Cambridge university press.
- Biber, Douglas & Johansson, Stig & Leech, Geoffrey Neil & Quirk, Randolph (eds.). 1999. *Longman grammar of spoken and written English*. 4th impr. Harlow: Pearson education.
- Biber, Douglas & Reppen, Randi & Schnur, Erin & Ghanem, Romy. 2016. On the (non)utility of Juilland's D to measure lexical dispersion in large corpora. *International Journal of Corpus Linguistics* 21(4). 439–464.
- Bybee, Joan. 1998. The emergent lexicon. *Chicago Linguistics Society* 34. 421–435.
- Chang, Jason J. S. & Chen, Shun-Der. 1995. The Postprocessing of Optical Character Recognition Based on Statistical Noisy Channel and Language Model. *Proceedings of the 10th Pacific Asia Conference on Language, Information and Computation*, 127–132. City University of Hong Kong, Hong Kong.
- Chen, Yu-Hua & Baker, Paul. 2010. Lexical Bundles in L1 and L2 Academic Writing. *Language Learning & Technology* 14(2). 30–49.
- Cheng, W. 2009. Income/interest/net : using internal criteria to determine the aboutness of a text. *Studies in Corpus Linguistics* 157–177.
- Cheng, Winnie & Greaves, Chris & Warren, Martin. 2006. From n-gram to skipgram to conogram. *International Journal of Corpus Linguistics* 11(4). 411–433.
- Choueika, Yaacov. 1988. Looking for needles in a haystack or: locating interesting expressions in large textual databases,. *RIAO 1988*, 609–624.
- Choueika, Yaacov & Klein, Timothy M. & Neuwitz, E. 1983. Automatic retrieval of frequent idiomatic and collocational expressions in a large corpus. *Journal for Literary and Linguistic Computing* 4. 34–38.
- Christiansen, Morten & Arnon, Inbal. 2017. More Than Words: The Role of Multiword Sequences in Language Learning and Use. *Topics in cognitive science* 9. 621–636.
- Church, Kenneth & Gale, William & Hanks, Patrick & Hindle, Don. 1991. Using Statistics in Lexical Analysis. *Lexical Acquisition: Exploiting On-line Resources to Build a Lexicon*, 115–164.
- Church, Kenneth & Gale, William & Hanks, Patrick & Hindle, Donald. 1989. Parsing, Word Associations and Typical Predicate-Argument Relations. *In Proceedings of the International Workshop on Parsing Technologies*, 103–112.
- Cortes, Viviana. 2002. Lexical bundles in Freshman composition. In Reppen, R. & Fitzmaurice, S. & Biber, D. (eds.), *Using Corpora to Explore Linguistic Variation* (Studies in Corpus Linguistics 9), 131–145. Amsterdam/Philadelphia:

- John Benjamins. (<https://benjamins.com/catalog/scl.9.09cor>) (Accessed May 19, 2021.)
- Cortes, Viviana. 2004. Lexical bundles in published and student disciplinary writing: Examples from history and biology. *English for Specific Purposes* 23. 397–423.
- Cowie, Anthony Paul. 1981. The Treatment of Collocations and Idioms in Learners' Dictionaries. *Applied Linguistics* II(3). 223–235.
- Cowie, Anthony Paul. 1994. Phraseology. In Asher, R.E. (ed.), *The Encyclopedia of Language and Linguistics*, 3168–3171. Oxford: Oxford University Press.
- Davies, Mark. 2016-present. English Corpora: most widely used online corpora. Billions of words of data: free online access. *English-Corpora.org*. (<https://www.english-corpora.org/corpora.asp>) (Accessed January 27, 2021.)
- De Cock, Sylvie. 2003. *Recurrent sequences of words in native speaker and advanced learner spoken and written English: a corpus-driven approach*. UCL - Université Catholique de Louvain.
- Dias, Gaël. 2002. *Extraction automatique d'associations lexicales à partir de corpora*. Lisbonne: Université Nouvelle de Lisbonne.
- Dias, Gaël. 2003. Multiword unit hybrid extraction. In *Workshop on Multiword Expressions of the 41st ACL meeting*, 41–48.
- Dias, Gaël & Guilleré, Sylvie & Bassano, Jean-claude & Gabriel, José & Lopes, José Gabriel Pereira. 2000. Combining Linguistics with Statistics for Multiword Term Extraction: A Fruitful Association? *RIAO '00: Content-Based Multimedia Information Access*, vol. 2, 1473–1491.
- Duan, Xuemin & Zan, Hongying & Bai, Xiaojing & Zahner, Christoph. 2020. Reusable Phrase Extraction Based on Syntactic Parsing. In Sun, Maosong & Li, Sujian & Zhang, Yue & Liu, Yang & He, Shizhu & Rao, Gaoqi (eds.), *Chinese Computational Linguistics*, 458–466. Cham: Springer International Publishing.
- Dunn, Jonathan. 2018. Multi-unit association measures: Moving beyond pairs of words. *International Journal of Corpus Linguistics* 23(2). 183–215.
- Dunning, Ted. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics* 19(1).
- Egbert, Jesse & Burch, Brent & Biber, Douglas. 2020. Lexical dispersion and corpus design. *International Journal of Corpus Linguistics* 25(1). 89–115.
- Ellis, Nick C. 1996. Sequencing in SLA: Phonological Memory, Chunking, and Points of Order. *Studies in Second Language Acquisition* 18(1). 91–126.
- Ellis, Nick C. 2002. Frequency effects in language processing: A Review with Implications for Theories of Implicit and Explicit Language Acquisition. *Studies in Second Language Acquisition* 24(2). 143–188.
- Ellis, Nick C. 2006. Language Acquisition as Rational Contingency Learning. *Applied Linguistics* 27(1). 1–24.

- Evert, Stefan. 2004. *UCS toolkit*. (<http://www.collocations.de/software.html#UCS>) (Accessed February 10, 2021.)
- Evert, Stefan & Krenn, Brigitte. 2005. Using small random samples for the manual evaluation of statistical association measures. *Computer Speech & Language* (Special Issue on Multiword Expression) 19(4). 450–466.
- Evert, Stefan & Uhrig, Peter & Bartsch, Sabine & Proisl, Thomas. 2017. E-VIEW-alation – a Large-scale Evaluation Study of Association Measures for Collocation Identification. *Proceedings of eLex 2017*, 531–549. Leiden, The Netherlands.
- Fazly, Afsaneh & Stevenson, Suzanne. 2006. Automatically Constructing a Lexicon of Verb Phrase Idiomatic Combinations. *Proceedings of the 11th Conference of the European Chapter of the ACL (EACL-2006)*, 337–344.
- Finlayson, Mark Alan & Kulkarni, Nidhi. 2011. Detecting Multi-Word Expressions improves Word Sense Disambiguation. *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World (MWE 2011)*, 20–24. Portland, Oregon, USA.
- Fletcher, William H. 2003-2011. “Phrases in English” - Home Page. (<http://phrasesinenglish.org/>) (Accessed March 1, 2020.)
- Fletcher, William H. 2004. Making the Web More Useful as a Source for Linguistic Corpora. In Connor, U. & Upton, T. (eds.), *Corpus Linguistics in North America 2002: Selections from the Fourth North American Symposium of the American Association for Applied Corpus Linguistics*. Amsterdam: Rodopi.
- Fletcher, William H. 2007. *KfNgram*. (<http://www.kwicfinder.com/kfNgram/>) (Accessed March 2, 2020.)
- Garner, James R. 2016. A phrase-frame approach to investigating phraseology in learner writing across proficiency levels. *International Journal of Learner Corpus Research* 2(1). 31–67.
- Götz, Sandra & Schilk, Marco. 2011. Formulaic sequences in spoken ENL, ESL and EFL: Focus on British English, Indian English and learner English of advanced German learners. In Mukherjee, Joybrato & Hundt, Marianne (eds.), *Exploring Second-language Varieties of English and Learner Englishes: Bridging a paradigm gap* (Studies in Corpus Linguistics 44), 79–100. Amsterdam: John Benjamins.
- Granger, Sylviane & Paquot, Magali. 2008. Disentangling the phraseological web. In Granger, Sylviane & Meunier, Fanny (eds.), *Phraseology*, 27–49. Amsterdam: John Benjamins Publishing Company.
- Granger, Sylviane & Paquot, Magali. 2010. Customising a general EAP dictionary to meet learner needs. In Granger, Sylviane (ed.), *ELexicography in the 21st Century: New Challenges, New Applications: Proceedings of ELex 2009, Louvain-la-Neuve, 22-24 October 2009*, 87–96. Presses univ. de Louvain.
- Greaves, Chris & Warren, Martin. 2007. Concgramming: A computer driven approach to learning the phraseology of English. *ReCALL* 19. 287–306.

- Gries, Stefan. 2008. Dispersions and adjusted frequencies in corpora. *International Journal of Corpus Linguistics* 13. 403–437.
- Gries, Stefan. 2009. Dispersions and adjusted frequencies in corpora: further explorations. *Language and Computers* 197–212.
- Gries, Stefan Th. 2013. 50-something years of work on collocations: What is or should be next *International Journal of Corpus Linguistics* 18(1). 137–166.
- Gries, Stefan. 2020. Analyzing dispersion. In Paquot, Magali & Gries, Stefan Th. (eds.), *Practical handbook of corpus linguistics*. Springer International Publishing.
- Gries, Stefan Th. & Durrant, Philip. 2020. Analyzing Co-occurrence Data. In Paquot, Magali & Gries, Stefan Th. (eds.), *A Practical Handbook of Corpus Linguistics*, 141–159. Springer International Publishing.
- Hyland, Ken. 2008a. As can be seen: Lexical bundles and disciplinary variation. *English for Specific Purposes* 27(1). 4–21.
- Hyland, Ken. 2008b. Academic clusters: Text patterning in published and postgraduate writing. *International Journal of Applied Linguistics* 18. 41–62.
- Hyland, Ken & Jiang, Feng (Kevin). 2018. Academic lexical bundles: How are they changing? *International Journal of Corpus Linguistics* 23(4). 383–407.
- Inurrieta, Uxo & Aduriz, Itziar & Ilaraza, Arantza Díaz de & Labaka, Gorka & Sarasola, Kepa. 2020. Learning about phraseology from corpora: A linguistically motivated approach for Multiword Expression identification. *PLOS ONE* 15(8). e0237767.
- Jespersen, Otto. 1924/1976. Living grammar. In *The philosophy of grammar*. London: George Allen & Unwin, 17–29. Reprinted in D.D. Bornstein (ed.) 1976. *Readings in the theory of grammar*. Cambridge, MA: Winthrop Publishers, 82–93.
- Joseph, Aurélie. 2013. Améliorer l'extraction et la description d'expressions polylexicales grâce aux règles transformationnelles. 42–55. Les Sables-d'Olonne.
- Juilland, Alphonse & Chang-Rodriguez, E. 1964. *Frequency dictionary of Spanish words* (The Romance Languages and Their Structures. Series 1 S1). The Hague: Mouton.
- Kjellmer, Goran. 1991. A mint of phrases. In Aijmer, Karin & Altenberg, Bengt (eds.), *English Corpus Linguistics*, 111–127. Longman. London.
- Kolesnikova, Olga. 2016. Survey of Word Co-occurrence Measures for Collocation Detection. *Computación y Sistemas* 20(3). 327–344.
- Kulmizev, Artur & Blankers, Bo & Bjerva, Johannes & Nissim, Malvina & van Noord, Gertjan & Plank, Barbara & Wieling, Martijn. 2017. The Power of Character N-grams in Native Language Identification. *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 382–389. Copenhagen, Denmark: Association for Computational Linguistics. (Accessed March 16, 2021.)

- Leech, Geoffrey N. 2011. Frequency, corpora and language learning. In De Cock, Sylvie & Gilquin, Gaëtanelle & Paquot, Magali (eds.), *A Taste for Corpora: In honour of Sylviane Granger* (Studies in Corpus Linguistics 45), 7–32.
- Lefer, Marie-Aude & Grabar, Natalia. 2015. N-grams in multilingual corpora: extracting and analyzing lexical bundles in contrastive studies.
- Lewis, Michael (ed.). 2000. *Teaching Collocation - Further Developments in the Lexical Approach*. Boston, Mass.: Thomson Heinle Language Teachi.
- Lin, Phoebe M. S. & Adolphs, Svenja. 2009. Sound Evidence: Phraseological Units in Spoken Corpora. In Barfield, Andy & Gylstad, Henrik (eds.), *Researching Collocations in Another Language: Multiple Interpretations*, 34–48.
- Mel'čuk, Igor. 2001. Collocations and Lexical Functions. In Cowie, Anthony Paul (ed.), *Phraseology: theory, analysis, and applications* (Oxford Linguistics), 23–53. Oxford: Oxford university press.
- Merkel, Magnus & Andersson, Mikael. 2000. Knowledge-lite extraction of multi-word units with language filters and entropy thresholds. *Proceedings of RIAO'2000*, 737–746.
- Meyer, David & Schvaneveldt, Roger & Ruddy, Margaret. 1975. Loci of Contextual Effects on Visual Word Recognition. In Rabbitt, P. M. A. & Dornic, S. (eds.), *Attention and performance V*, vol. 5, 98–118. Academic Press. London.
- Michelbacher, Lukas & Evert, Stefan & Schutze, Hinrich. 2007. Asymmetric Association Measures. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, 6. Borovets, Bulgaria.
- Monti, Johanna & Arčan, Mihael & Sangati, Federico. 2020. Translation asymmetries of multiword expressions in machine translation: An analysis of the TED-MWE corpus. In Corpas Pastor, Gloria & Colson, Jean-Pierre (eds.), *Computational Phraseology* (IVITRA Research in Linguistics and Literature), vol. 24, 24–41. Amsterdam/Philadelphia: John Benjamins Publishing Company.
- Moon, Rosamund. 1997. In Schmitt, Norbert & McCarthy, Michael (eds.), *Vocabulary: description, acquisition and pedagogy* (Cambridge Language Teaching Library), 40–63. Cambridge: Cambridge university press.
- Nerima, Luka & Seretan, Violeta & Wehrli, Eric. 2006. Le problème des collocations en TAL. *Nouveaux cahiers de linguistique française* 27. 95–115.
- Nesselhauf, Nadja. 2005. *Collocations in a learner corpus* (Studies in Corpus Linguistics 14). Amsterdam: Benjamins.
- O'Donnell, Matthew & Römer, Ute & Ellis, Nick. 2013. The development of Formulaic sequences in first and second language writing: Investigating effects of frequency, association, and native norm. *International Journal of Corpus Linguistics* 18(1). 83–108.

- O'donnell, Matthew Brook. 2011. The adjusted frequency list: A method to produce clustersensitive frequency lists. *International Computer Archive of Modern and Medieval English* 35. 135–169.
- Palkovic, Aj. 2008. Improving Optical Character Recognition. *Proceedings of the 2nd Villanova University Undergraduate Computer Science Research Symposium (CSRS 2008)*.
- Pan, Fan & Reppen, Randi & Biber, Douglas. 2020. Methodological issues in contrastive lexical bundle research. *International Journal of Corpus Linguistics*. John Benjamins Publishing Company 25(2). 214–228.
- Pawley, Andrew & Syder, F. 1983. Two puzzles for linguistic theory: Nativelike selection and nativelike fluency. In Richards, J. C. & Schmidt, R. W. (eds.), *Language and Communication*, 191–225. London: Longman.
- Pecina, Pavel. 2005. An extensive empirical study of collocation extraction methods. *Proceedings of the ACL Student Research Workshop on - ACL '05*, 13–18. Ann Arbor, Michigan.
- Pecina, Pavel & Schlesinger, Pavel. 2006. Combining Association Measures for Collocation Extraction. *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 651–658. Sydney.
- Pedersen, Ted. 1998. Dependent bigram identification. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1197.
- Pedersen, Ted & Banerjee, Satanjeev & McInnes, Bridget T. & Kohli, Saiyam & Joshi, Mahesh & Liu, Ying. 2011. The Ngram Statistics Package (Text::NSP) - A Flexible Tool for IdentifyingNgrams, Collocations, and Word Associations. *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World (MWE 2011)*, 131–133. Portland, Oregon, USA.
- Peters, Ann M. 1983. *Units of language acquisition*. Cambridge University Press. Cambridge.
- Ramisch, Carlos. 2012. Une plate-forme générique et ouverte pour le traitement des expressions polylexicales (An Open and Generic Framework for the Acquisition of Multiword Expressions) [in French]. *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 3: RECITAL*, 137–149. Grenoble, France: ATALA/AFCP.
- Ramisch, Carlos. 2020. Computational phraseology discovery in corpora with the MWETOOLKIT. In Corpas Pastor, Gloria & Colson, Jean-Pierre (eds.), *Computational Phraseology* (IVITRA Research in Linguistics and Literature), vol. 24, 111–133. Amsterdam: John Benjamins Publishing Company.
- Renouf, Antoinette & Sinclair, John McHardy. 1991. Collocational frameworks in English. In Aijmer, Karin & Altenberg, Bengt (eds.), *English corpus linguistics: studies in honour of Jan Svartvik*, 128–143.
- Römer, Ute. 2010. Establishing the phraseological profile of a text type: The construction of meaning in academic book reviews. *English Text Construction* 3(1). 95–119.

- Sag, Ivan A. & Baldwin, Timothy & Bond, Francis & Copestake, Ann & Flickinger, Dan. 2002. Multiword Expressions: A Pain in the Neck for NLP. In Gelbukh, Alexander (ed.), *Computational Linguistics and Intelligent Text Processing* (Lecture Notes in Computer Science), 1–15. Berlin, Heidelberg: Springer.
- Saito, Kazuya. 2020. Multi- or Single-Word Units? The Role of Collocation Use in Comprehensible and Contextually Appropriate Second Language Speech. *Language Learning* 70(2). 548–588.
- Schmid, Helmut. 1995. Improvements in Part-of-Speech Tagging with an Application to German. *Proceedings of the ACL SIGDAT-Workshop*, 13–25. Dublin, Ireland.
- Scott, Mike. 2020. *WordSmith Tools version 8*. Stroud: Lexical Analysis Software. (<https://lexically.net/wordsmith/>) (Accessed August 18, 2020.)
- Scott, Mike & Tribble, Chris. 2006. *Textual patterns: key words and corpus analysis in language education* (Studies in Corpus Linguistics 22). Vol. 11 (2). Amsterdam: Benjamins.
- Shannon, C. E. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal* 27. 379–423, 623–656.
- Shimohata, Sayori & Sugio, Toshiyuki & Nagata, Junji. 1997. Retrieving Collocations by Co-Occurrences and Word Order Constraints. *Computational Intelligence* 15(2). 476–481.
- Silva, Joaquim Ferreira Da & Lopes, Gabriel Pereira & Torre, Quinta Da & Caparica, Monte Da. 1999. A local Maxima Method and a Fair Dispersion Normalization for Extracting Multiword Units. In *Proceedings of the 6th Meeting on the Mathematics of Language*, 369–381.
- Simpson-Vlach, Rita & Ellis, Nick C. 2010. An Academic Formulas List: New Methods in Phraseology Research. *Applied Linguistics* 31(4). 487–512.
- Sinclair, J. 1991. *Corpus, Concordance, Collocation*. Oxford, UK: Oxford University Press.
- Smadja, Frank. 1993. Retrieving Collocations from Text: Xtract. *Computational Linguistics* 19. 143–177.
- Smadja, Frank & McKeown, Kathleen. 1991. Using collocations for language generation. *Computational Intelligence* 7(4). 229–239.
- Stubbs, Michael & Barth, Isabel. 2003. Using recurrent phrases as text-type discriminators: A quantitative method and some findings. *Functions of Language* 10(1). 61–104.
- Tremblay, Antoine & Derwing, Bruce & Libben, Gary & Westbury, Chris. 2011. Processing Advantages of Lexical Bundles: Evidence From Self-Paced Reading and Sentence Recall Tasks. *Language Learning* 61(2). 569–613.
- Verkoeijen, Peter P.J.L. & Rikers, Remy M.J.P. & Schmidt, Henk G. 2004. Detrimental Influence of Contextual Change on Spacing Effects in Free Recall. *Journal of Experimental Psychology. Learning, Memory & Cognition* 30(4). 796–800.

- Wahl, Alexander & Gries, Stefan. 2020. Computational extraction of formulaic sequences from corpora: Two case studies of a new extraction algorithm. In Corpas Pastor, Gloria & Colson, Jean-Pierre (eds.), *Computational and corpus-based phraseology* (IVITRA Research in Linguistics and Literature), vol. 24, 84–110. Amsterdam: John Benjamins.
- Watrin, Patrick. 2007. Collocations et traitement automatique des langues. 26ème *Colloque international sur le lexique et la grammaire (LGC'07)*, 191–198. France.
- West, Michael (ed.). 1953. *General Service List of English Words*. Harlow: Longman.
- Wible, David & Kuo, Chin-Hwa & Chen, Meng & Tsao, Nai-Lung & Hung, Tsung-Fu. 2006. A Computational Approach to the Discovery and Representation of Lexical Chunks. *The 13th Conference on Natural Language Processing (TALN 2006)*, 868–875. Leuven, Belgium.
- Wiechmann, Daniel. 2008. On the computation of construction strength: Testing measures of association as expressions of lexical bias. *Corpus Linguistics and Linguistic Theory* 4(2). 253–290.
- Wray, Alison. 2002. *Formulaic Language and the Lexicon*. Cambridge: Cambridge University Press.
- Yang, Shouxu. 2003. Machine Learning for collocation identification. *International Conference on Natural Language Processing and Knowledge Engineering Proceedings (NPL-KE)*, 315–320. Beijing.

