

Gabriel 1902228 Project Report

by Gabriel Atareyele Avbenake

FILE	17802_GABRIEL_ATAREYELE_AVBENAKE_GABRIEL_1902228_PROJECT_REPO RT_260366_1379235004.DOCX (1.35M)		
TIME SUBMITTED	19-NOV-2020 10:13PM (UTC+0000)	WORD COUNT	20988
SUBMISSION ID	136841261	CHARACTER COUNT	128422



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

A Design of a Nominal MWE Tool

by

Gabriel Avbenake

Supervisor: Nitheesh Kaliyamurthy

Project submitted as part of the requirements
for the award of M.Sc. Applied Computing

November 2020

DECLARATION

I, Gabriel Avbenake, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : 

Date : 17th of November, 2020

FORM OF CONSENT

I, Gabriel Avbenake, hereby consent that my Project, submitted in candidature for the *M.Sc. Applied Computing* degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : 

Date : 17th of November, 2020

Abstract

Multiword expressions (MWEs) are word combinations that present some peculiarities in language, behaving as single words to some extent, especially at the syntactic, semantic and pragmatic level of language. Natural Language Processing (NLP) needs to deal with MWEs because they are ubiquitous in language. Ignoring MWEs in NLP tasks generate ungrammatical, ambiguous or unnatural output leading to probable information loss.

Several studies have investigated the discovery of MWEs automatically from texts, but failed to incorporate the extracted MWEs in a bid to influence the NLP applications. Subsequently, in this study an attempt was made to discover MWE and the extracted MWEs were lexically encoded in a format that could readily be employed in NLP applications. Such appropriately built electronic MWE lexicons are important for NLP, since MWEs behave like single words and need to be dealt with individually.

Therefore, using a positivist approach alongside deductive and quantitative research method with Waterfall SDLC model, employing HTML and CSS for frontend, with Python, PHP and MySQL for backend, a web-based tool was built to extract Nominal MWEs from BNC texts. The two human evaluators deduced a precision rate ranged from 0.39 to 0.75 and a recall rate ranged from 0.64 to 0.96 when they validated the MWEs which were presented in NLP-ready lexicon that is easily converted to XML-LMF format. The tool was further designed to improve its extraction ability with time utilizing all past human evaluations carried out with it, therefore speeding up the work of lexicographers or MWE lexicon compilers.

Acknowledgement

I am grateful to all my teachers in the past that brought me this far. I would like to thank in a special way my lecturers at UWTSD, especially Gaynor Thomas, Nik Whitehead and others.

My special gratitude to my supervisor, Nitheesh Kaliyamurthy, for his support, encouragement, advice and insights. I am very grateful to the two evaluators; whose efforts were integral to this research.

Special thanks to the Dominican Province of Nigeria and Ghana for believing in me and supporting me.

My family and friends, your support is very much appreciated.

Table of Contents

DECLARATION.....	ii
FORM OF CONSENT.....	ii
Abstract	iii
Acknowledgement	iv
Table of Contents	v
Table of Figures	vii
Chapter One: Introduction.....	8
1.1 Background to the Proposed Research	8
1.2 Research Problem.....	8
1.3 Rationale for the Research	10
1.4 Scope of Study	10
1.5 Aims and Objectives	11
1.6 Expected Outcomes.....	11
1.7 Limitation of the Study.....	11
Chapter Two: Literature Review	13
2.1 NLP needs MWE Lexicons.....	13
2.2 The Quest for an Appropriate MWE Lexical Encoding for NLP	13
2.2.1 The Complexities of MWE Lexical Encoding.....	15
2.2.2 Between Fixed and Flexible MWE Lexical Encoding.....	16
2.3 Identifying Multiword Expression (MWE)	17
2.3.1 Approaches to Automatic MWE Discovery	19
2.3.2 SpaCy: An Impressive Pre-processing Tool.....	23
2.3.3 Evaluating MWE Discovery Process.....	24
Chapter Three: Research Methodology	27
3.1 Introduction.....	27
3.2 Research Philosophy.....	27
3.3 Research Approach.....	27
3.4 Quantitative Methodology	28
3.5 Procedures and Techniques	28
3.5.1 Pre-processing.....	29
3.5.2 Extraction	29
3.5.3 Evaluation.....	30
3.5.4 Lexical Encoding	30

3.6 Conclusion	31
Chapter Four: Design and Implementation.....	32
4.1 Introduction.....	32
4.2 Waterfall SDLC Model	32
4.2.1 Requirement specification and analysis.....	32
4.2.2 Designing project architecture: A modular architectural design	33
4.3 Design	33
4.4 Implementation	37
4.4.1 Generating Frequency Table	43
4.4.2 Database and Tables	47
4.4.3 Setting Up Python.....	49
Chapter Five: Analysis and Evaluation	50
5.1 Introduction.....	50
5.2 Results and Discussion.....	50
5.3 Conclusion	52
Chapter Six: Conclusions & Recommendations	54
6.1 Summary.....	54
6.2 Conclusion	55
6.3 Recommendations for Further Research	56
Chapter Seven: Reflection	57
References.....	58
APPENDICES	65

Table of Figures

Figure 1: Modules of the Design Architecture	33
Figure 2: Flowchart of the Extraction Section	34
Figure 3: Flowchart of the Extraction section (continuation)	35
Figure 4: Flowchart for MWE Evaluation	36
Figure 5: A Skeletal Presentation of the Workflow	37
Figure 6: Web Interface of the Tool	38
Figure 7: Feedback Interface for Pre-Processing	39
Figure 8: Syntactic Extraction Feedback Interface	40
Figure 9: Statistical Extraction Feedback Interface	41
Figure 10: Human Evaluation Interface.....	42
Figure 11: A snippet of an XML file in the BNC	44
Figure 12: Code to extract text from BNC XML files	45
Figure 13: Code to extract and save frequency table from BNC text	46
Figure 14: Code to extract and save frequency of Bigrams	46
Figure 15: Description of the mwes Table	48

Chapter One: Introduction

1.1 Background to the Proposed Research

The field of Natural Language Processing (NLP) has developed rapidly over recent decades. NLP is a branch of computer science that enables computers to extract meaning from unstructured text [1] [2]. NLP is important because it fosters communication, knowledge management and discovery [3].

Some applications of NLP includes but are not limited to Machine Translation (MT), Information Retrieval (IR) especially web search, Optical Character Recognition (OCR), digital assistants, Spell checking and grammar corrections, morphological and syntactic analysis (e.g. POS taggers and Parsers) and other computational semantics (e.g. question-answering, automatic summarisation, text simplification) [4]. These have greatly impacted our lives in this information age era.

Multiword expressions (MWEs) are part and parcel of what NLP has to deal with. MWEs are replete in our everyday speech and writings. Idioms such as he kicked the bucket, Shoot the breeze, rack one's brains and so on are some examples of MWEs. Apart from idiomatic expressions, many other word combinations present some peculiarities which allow us to see them as MWEs [5]. MWEs can occur in a wide range of lexical and syntactic configurations such as nominal, verbal MWEs and so on.

1.2 Research Problem

In a very simplistic form, linguistics deal with individual words as well as the rules for combining these words known as grammar. However, there are some constructs that are more than one word but behave/function as a single word to some extent. They do not behave exactly like other group of words in the language, such as phrases, clauses or sentences [6]. Though MWEs are quite often understood by the native speakers of a language, what constitute MWEs and what does not is controversial [7]. It is paramount that the features of MWEs are to be determined, if their identification, extraction and treatment in NLP applications are going to be possible.

Several studies have been carried out on the discovery of MWEs automatically from texts [8] [9] [10] [11] [12]. However, most of these MWE discovery efforts have simply

ended in extracting a list of MWEs from corpora; these efforts have often not been put back into the bigger picture of influencing NLP applications [6]. As Constant et al., 2017 submitted in their survey: “it remains unclear whether discovery is required or even useful to support and/or replace lexicographic work in production scenarios” [6]. Therefore, MWE discovery efforts would be more useful if at least the extracted MWEs are linked to lexicons or put in lexicon-format for future use by NLP applications. In other words, it would be more useful if the extracted MWEs are encoded lexically [13] [14]. Such lexically-encoded extracted MWEs could then be added to enrich existing lexicons after verification. In encoding the extracted MWEs, it is essential that a standard MWE lexicon format is used so that NLP applications can easily be adapted to use it [14].

Consequently, this study is an attempt to build a tool (web-based and visual) that would extract Nominal MWEs from corpora and put them in a MWE lexicon (in a format that is NLP-ready), linking to existing lexicons to retrieve their meaning, if available. To be clear, this is not an attempt at a new (inventive) method, but an attempt to use the available state-of-the-art to design a tool that would achieve the best results at automatically discovering MWE and putting them in a lexicon-entry state (with their useful features illustrated) such that NLP could readily use them to improve accuracy and efficiency.

This is in line with objectives of the 2020 SIGLEX-MWE Workshop on Multiword Expressions, tagged MWE-LEX 2020, which is a joint workshop on multiword expressions and electronic lexicons [15]. The organizers admit that it is paramount:

- That MWEs be discovered automatically in texts using justifiable and verifiable principles
- That such automatically extracted MWEs be put in NLP-oriented MWE lexicon format such that it can be readily plugged into NLP applications to improve efficiency and accuracy.

This study is a modest attempt at these two connected tasks.

1.3 Rationale for the Research

Building NLP applications that are capable of dealing with MWEs is very important because:

- (1) MWEs are ubiquitous in everyday language [16] [17]
- (2) Ignoring MWEs in NLP tasks generate ungrammatical, ambiguous or unnatural output [5] leading to probable information loss.

Despite their prevalence, MWEs have often been neglected in the development of NLP applications. A quick search for the French translation of the English statement “He kicked the bucket yesterday” yielded “Il a lancé le seau hier” on Google Translate (<https://translate.google.com>) and “Il a donné un coup de pied au seau hier” on DeepL (<https://www.deepl.com/en/translator>), on PROMT.One (<https://www.online-translator.com/>), on Collins Translator (<https://www.collinsdictionary.com/translator>) and on Translate.com (<https://www.translate.com/>). This was carried out on the 25th of Sept, 2020. These translations are all wrong because they are attempting a literal translation. They are oblivious of the presence of a MWE that changes everything. The correct translation should have been: “il est mort hier”.

This study would not simply attempt MWE discovery that has been the focus of many researches in the past; it would ensure that the extracted MWEs are lexically encoded in a format that can readily be used in NLP application. In other words, the extracted MWEs could be used to augment existing lexicon to improve the accuracy of NLP applications. Such appropriately built electronic MWE lexicons are important for NLP [15] [18].

1.4 Scope of Study

MWEs display varied linguistic properties and idiosyncrasies, as such automatic MWE extraction from corpora may require different methods for different types of MWEs [19]. Given the time constraint of this study, the focus would not be all the various types of MWEs. The scope of this study would be limited to only Nominal MWEs. This is one of the groups in the classifications of MWEs, as would be discussed later.

1.5 Aims and Objectives

The aim of this research is to design a hybrid system that can extract Nominal MWEs from English texts, link them to corresponding entries in lexicons and augment existing resources with possible new expressions.

The objectives are:

- To design a tool which will automatically discover Nominal MWEs in large non-annotated English corpora to enrich MWE lists from traditional human-readable lexicons.
- To evaluate how useful the tool is, in terms of accuracy. Human evaluators will be used.
- To ensure the possibility of easy integration of the enriched lexicon into NLP applications.

1.6 Expected Outcomes

At the end of the project, it is expected that there would be a tool that will produce an enriched MWE lexicon, with minimal human effort, from various corpora. The enriched MWE lexicon would be in a format ready to be plugged into the pipeline of various NLP applications.

This would improve NLP applications to generate less ambiguous, more natural and grammatical output to prevent information loss and boost confidence in NLP output. It would ultimately foster communication, knowledge management and discovery.

1.7 Limitation of the Study

Apart from time constraint, one of the most significant limitations of this study would be lack of expert context knowledge, since the researcher is not an expert in the area of lexicography and linguistics. An expert in linguistics or bi-lingual would prove advantageous. Although the study would be carried out with lots of readings of published linguistic articles, efforts would be made to gain more knowledge through self-learning. This would, certainly, impact on the timescale.

Another limitation would be at the evaluation phases. It would have been ideal to have as many as 5-8 qualified human evaluators in order to get consensus where there are disagreements in their labelling, which would compensate for human error and biases. However, given time constraint and availability of willing experts, evaluators might be limited to two or three.

Chapter Two: Literature Review

2.1 NLP needs MWE Lexicons

Generally, NLP applications help computers to interpret and deal with human language [20]. Human language system deals with phonetics, phonology, morphology, syntax, semantics and pragmatics. Phonetics and phonology are associated with sounds. Morphology, on the other hand, is the study of the structure of words and other meaningful units of language such as prefix, suffix, and other morphemes. Syntax is the set of rules that govern how words and other morphemes are combined to form valid larger units like phrases, clauses and sentences. Semantics deals with the literal meanings of words and combinations of words. Whereas pragmatics studies sentence meaning in context or language usage. A combination of these corroborates the fact that every language has lexicon and grammar. Lexicon is a collection of all words in a language including their properties and meaning, while grammar is the set of rules in a language for creating words and sentences [21]. Consequently, lexicons help humans and NLP applications to identify and understand the various words in a language and their respective properties.

In a very simplistic form, one could say that language systems deal with words as well as the rules for combining them. However, there are some constructs that are greater than one word but behave as a single word to some extent. These do not behave exactly like other group of words in the language, such as phrases, clauses or sentences. For this reason, MWEs do not always follow the rules of grammar, hence cannot be dealt with at that level. It was established that MWE display lexical, syntactic, semantic, pragmatic and/or statistical idiomaticity [22]. They are usually non-compositional in their semantics and syntax, and therefore, ought to be captured in lexicons with their properties to enable NLPs deal with them better. In any case, MWE lexicons improve the performance of NLP, a view shared by Savary et al., 2019 that carried out specific analysis in this area [14]. Several other researchers share in this position [18] [15] [13] [23].

2.2 The Quest for an Appropriate MWE Lexical Encoding for NLP

In generating MWE lexicons, the entries should be in a format that is NLP-compliant. In other words, it should be encoded such that would help NLP easily employ it to

improve its accuracy and efficiency. Thus, it should contain the characteristic features of MWE that NLP need to be aware of in processing MWE accurately and efficiently.

However, in developing these lexicons, the question of what specific properties that should be included and how they should be represented are yet to be standardized research-wise. Defining such standard MWE lexicon format is one of the objectives of MWE-LEX 2020: “the issues of lemmatization and of standardised representation of morphological, syntactic and semantic properties of MWEs” [15]. Many NLPs do not use MWE-lexicons today because they lack unified format which makes their integration difficult [14].

Attempts have been made to develop a standard for encoding MWEs, for example the ISLE project within EAGLES [24], the XMELLT project [25] and the PARSEME [26]. Although, a multilingual framework meant to unify annotation methodology for Verbal MWEs have previously been presented [26], the researchers submitted that the framework performance varies greatly among languages and systems. Besides, their MWEs did not include features such as variability, discontinuities, and so on.

Villavicencio and co-workers proposed their encoding architecture in the form of a database, with the flexible MWEs requiring a 3-tier table system. The first and second has the definitions of the idiomatic elements and MWE components respectively, while the third has the specification of a class for the MWE [27]. Meanwhile, another researcher discussed the various issues in using a relational database to store MWEs lexical entries [28], and others have used a treebank structure in representing MWE, like in the Prague Dependency Treebank, RoRefTrees Romanian Treebank, and so on [29].

In spite of all these attempts, there is no general standard accepted by linguistics. What is required is MWEs representation that is implementation-independent as possible but from which representations, specific to particular NLP applications can be derived in a maximally automated manner [30]. Probably, such a model could only result from a consolidated effort by various linguistic experts drawn from various domains and languages. This is why MWE-LEX 2020 is a step in the right direction, hoped to pave the way towards a better understanding of the issues of standardized MWE lexicon, and subsequently foster convergence and scientific innovation [15].

2.2.1 The Complexities of MWE Lexical Encoding

Formulating such uniform standards is no mean feat. The composite and varied nature of MWEs suggest a very complex formalism which could be difficult to understand or accommodate [14] [31]. Also, the encoding properties could be in conflict with each other, since the lexical encoding format would have to relate with three interfaces with their different requirements [31]. These interfaces include: the Lexical Object, Human User and Lexical Resource.

The lexical object is the object to be encoded, in this case, the MWEs. With respect to the lexical object, the encoding format has to be complete, concise and consistent [31]. Generally, the encoding is complete when every properties of the lexical object, in this case MWE, is encoded in the format and should represent the following [14] [31]:

- the morphological categories and syntactic structures of the MWE itself
- the rule-based combinatorial description of its variability patterns
- information about their constituent elements and their features, for example, their lemmas and parts of speech and so on [32]
- the meaning of the MWE
- possibly, some example sentences where it is used.

It is evident that such complete format would be arduous, if not gruelling. Consequently, Savary and co-workers opined that “such lexicons need not necessarily achieve a linguistically complete modelling of MWEs’ behaviour, but they should provide minimal morphosyntactic information to cover some potential uses” [14]. The trouble is that these potential uses might vary from one NLP app to another. Therefore, the minimal morphosyntactic information needed by one NLP app might be different from those needed by another NLP app.

Furthermore, the format is concise if the encoding is as brief as possible without superfluous details, yet incorporating all important information. Finally, it is consistent if it is always behaving in the same way, that is, it is constantly adhering to the same principles, having regular ways of expressing the same features [31].

For the human user, the encoding should be transparent, flexible and sufficiently powerful to capture generalizations [31]. Basically, the encoding is transparent if it is easy for the human user to understand and remember what its code stands for,

otherwise, the user would have a hard time processing it and might misrepresent it. Similarly, it is flexible if it allows the user to encode previously unforeseen properties or adapt the code as properties change over time. More so, these new encodings need to be generalized to avoid redundancy and encodings should be reusable.

Lexical resource is an electronic representation of lexical encodings that can be directly used in NLP application [31]. With respect to that, the encoding format would be dealing with many possible lexical resources, since NLP application can vary in their interface specifications. Accordingly, the encoding format would have to be versatile such that electronic conversion to various lexical resources interface should be feasible, fast and without errors [31].

2.2.2 Between Fixed and Flexible MWE Lexical Encoding

Moreover, several MWE encoding format have been proposed. Some of them are fixed such that *'their encoding language (basically the set of property names and their interpretation) cannot be freely chosen or extended'* [31]. Others are fully flexible encoding formats, in that property name can be freely chosen, with the implication that there is more than one way of implementing an object within such formats. It has been shown that flexible encoding formats tend to do better than fixed formats because shared properties of MWEs can be unforeseen, hence listing them all in advance is difficult [31].

Recently, Lichte and co-workers analysed four encoding formats (DuELME, Walenty, PATR-II and XMG). While DuELME and Walenty are fixed encoding format, PATR-II and XMG are fully flexible. They observed that the fully flexible encoding format were superior in terms of transparency and flexibility, but performed poorly in terms of electronic versatility and implementation friendliness. In a seemingly contradictory context, the fixed formats were superior in terms of electronic versatility and implementation friendliness, but performed badly in terms of transparency and flexibility [31].

In the current study, a semi-flexible encoding would be employed. It would exploit Universal Dependencies (UD) annotations for encoding parts of speech, morphological features, and syntactic dependencies [33]. This semi-flexible encoding would be designed with a possibility of easy conversion to a full XML-encoded lexicon [18], that

conforms to the LMF standard [34] [35]. The encoding would contain three core elements as follows; the description of the MWE itself (part of speech, lemma, contiguous capability , and so on), a description of its components (their lemma and parts of speech) and the combination pattern described using UD relations. In addition, there would be two optional components vis-a-vis the meaning of the MWE and sample sentences where the MWE is used. The former would be included where its meaning could be extracted from lexicons where they are pre-existent, particularly WordNet and BabelNet and likewise the latter. Where the MWE meaning is found on WordNet, the WordNet *synset* information could be used to enable the automatic retrieval of synonyms, and so on.

2.3 Identifying Multiword Expression (MWE)

Some of the properties of MWE that should be lexical encoded, as discussed in section 2.2.2, are the same properties exploited during some MWE discovery methods. As a result, MWE discovery could aid MWE encoding. However, automatic discovery of MWE is not easy because of the pervasive and complex nature of MWEs in language [7], especially the diversity of their semantic, lexical, statistical and syntactical properties. The various methods currently practiced for MWE discovery exploit the several features of MWEs that distinguish them from other non-MWE word combinations. These features include, but are not limited to the following:

Collocation / Statistical Idiomatcity: Prominent among the features of MWEs is their characteristic two or more words that often exist together [17]. This co-occurrence is beyond coincidence. They are marked with high frequency relative to the component words or alternatives of the expression [22]. For example; flawless, immaculate, and spotless are strong synonyms of impeccable. However, it is 'impeccable' that is often used with 'credentials' to a higher frequency than the other synonyms. Similarly, one would say '*fast food*' rather than '*quick food*', but '*a quick meal*' rather than '*a fast meal*'. In this co-occurrence, non-substitutability of synonym words is one feature that MWE display.

Since collocation is a feature that most MWEs share, various statistical association measures can be used to calculate the frequency of their co-occurrence as means to identify these MWEs. This phenomenon would be discussed in the ensuing section.

Non-compositionality: This is a complex term that could have different implications [17]. At a basic level, it implies that the meaning of the entire expression is not determined by or derived entirely from the meanings of its constituent units [36]. This is also known as Semantic Idiomaticity [22], and It ranges from partial to holistic non-compositionality [37]. For example, *Spill the beans* and *Ivory tower* have partial non-compositionality, whereas *Kick the bucket* and *Hot dog* have holistic non-compositionality.

Non-compositionality can equally be understood from the syntactic level. This occurs when the syntax of a term is not derived directly from its component unit [22], otherwise known as Syntactic Idiomaticity. For example, '*by and large*' is an adverb, but its components: 'by' is a preposition, 'and' is a conjunction and 'large' is an adjective. How does preposition + conjunction + adjective = adverb? Syntactic anomaly of the part-of-speech (POS) is a regular feature displayed by MWEs. Consequently, the part of speech of each MWE has to be understood on an individual basis. This is where the various categories of MWEs that would be discussed in the ensuing section comes into play.

Ambiguity: This is another feature of MWE referring to the tendency to be open to both compositional and MWE reading depending on the context [6]. For example, the reading of '*by the way*' in the following sentences varies:

- She picked some wild flowers *by the way*. – means ON THE ROAD; A compositional, non-MWE reading
- She was impressed *by the way* he handled the situation. – means HOW
- By the way I replied him, he must have suspected something is wrong. – means FROM HOW
- By the way, I have told him already. – means INCIDENTALLY or IN ADDITION.

Baldwin and Kim referred to this phenomenon as Pragmatic Idiomaticity; 'the condition of MWE being associated with a fixed set of situations or a particular context' [22]. For example, '*Good morning*' could be a form of greeting, but in this sentence and context: '*He had a good morning*', it is not a greeting; it means a pleasant morning. Also, Jack drank a Red Bull this morning; Red Bull here is an MWE, the name of a drink. Whereas, the red bull won the bull fight this morning; red bull here is not an MWE, it is simply a description of a bull by its pigmentation.

Classification-based features: MWEs are heterogenous, are of various types and exhibit some category-based features. In literature, there are different classifications using various schemes [22] [17]. Without getting involved in all the arguments, it is worth noting that, morpho-syntactically, MWEs are either fixed, semi-fixed, or flexible.

Fixed MWEs show some degree of fixedness in terms of structure and do not undergo any morphological modifications or word-order alterations [22]. They include function words such as *by and large*, *ad hoc*, *as well as*, *by the way*, etc; while other times functioning as prepositions, adverbs, adjectives or conjunctions.

Semi-Fixed MWEs could undergo some morphological variations in the absence of word-order alterations [22]. For example, ‘kick the bucket’ and ‘kicked the bucket’ are all correct MWEs, while ‘the bucket was kicked’ is no longer an MWE. Furthermore, Semi-Fixed MWEs consists of non-decomposable idioms and Nominal MWEs.

Syntactically flexible MWEs undergo both morphological variations and word-order alterations [22]. Particularly, they could be non-congruous, while possessing some foreign words in-between their components [6]. For example, whereas ‘put on’ could be non-congruous, as in the sentence: *She puts her red dress on every Thursday*; ‘hot dog’ is usually congruous. In addition, syntactically flexible MWEs include Verb-Particle Construction (VPC) (such as *find out*, *make out*, etc), Light-Verb Constructions (LVC) (such as *give confidence*, *make a living*, etc) and Decomposable idioms. Subsequently, in identifying such non-congruous MWEs, the core elements have to be recognized even if they are scattered apart by some alien words. In any event, Nominal MWEs which is the focus of this research does not display non-contiguity.

2.3.1 Approaches to Automatic MWE Discovery

A plethora of approaches have been used to identify MWEs in corpora. These approaches leverage on various MWE features that were discussed earlier. Considering MWEs display varied linguistic properties and idiosyncrasies, evidently, automatic MWE extraction from corpora may require different methods for the various types of MWEs [19]. Indeed, myriads of approaches abound in literature [8] [9] [10] [11] [12].

Admittedly, these approaches are usually grouped according to the methodology adopted, and are broadly categorized in three cases; Linguistic, Statistical and Hybrid Approaches.

2.3.1.1 Linguistic Approaches

Generally, linguistic approaches are methodologies that attempt to identify MWEs using linguistic properties, such as syntactic and morphological or semantic structures. These approaches use linguistic filtering techniques aimed at recognizing specific syntactic patterns associated with certain MWEs or semantic interpretations, context analysis or translations related to certain MWEs [38].

Linguistic approaches could either use semantic patterns only, syntactic and morphological patterns only or a combination of semantic-syntactic-morphological patterns. On the one hand, the semantic style exploits the non-compositionality features of MWEs by using various semantic patterns such as executing context analysis or checking translations by comparing parallel corpora in two or more languages. Such methods tend to better handle the ambiguity property of MWE since the contexts of the text is being taken into consideration.

Lately, Yazdani and co-workers employed an unsupervised learning method to investigate a range of functions that can model semantic composition of English noun compounds [36]. They used word embedding to represent words and compounds in distributional vector-space models that represent the semantics of words and phrases. Subsequently, these vector-space models were then passed to various composition functions to model compositionality and concluded that although the quadratic polynomial projection and neural networks are complex with several adjustments, they model semantic composition more effectively. Their objective was to identify non-compositionality, that is, if a particular English noun compound cannot be modelled with a relatively low error using their semantic composition function, such compound would be considered non-compositional and a MWE candidate. Similarly, additive and multiplicative composition functions with similarity-based models have been used to measure compositionality [39].

On the other hand, the syntactic style used the syntactic patterns of MWEs to identify them by leveraging on the fact that certain types of MWEs have some distinct syntactical structures [38]. For instance, LVCs usually comprise of a Verb with a Noun or Prepositional phrase, VPCs usually comprise of a Verb plus one or more particles,

and so on. Certainly, not all word combinations that have such syntactic structures are MWEs. It has been established that several methods can be used to identify true MWE candidates, and while some use semantic measures others use statistical measures.

The possibility of syntactic pattern identification is preceded by certain pre-processing including; Part-of-speech (POS) tagging, lemmatization and syntactic parsing. Therefore, using the desired syntactic structure, rules are constructed using regular expressions or similar language to search the text for desired patterns such as *[noun, noun]or[adjective, noun]* and so on. Usually, at the end, a list of unwanted words, known as stop-list, is employed to eliminate false MWE candidates and could be ubiquitous simple collocations such as “some persons” or “that boy”, even if they are in the syntactic form *[adjective, noun]*. Sometimes, this list is generated automatically from corpora eliminating words with very high frequency or generic function words [38].

2.3.1.2 Statistical Approaches

Since collocation is one of the prominent properties of MWE, various statistical association measures have been proposed to evaluate and determine the frequency of co-occurrence of certain terms as well as compute their degree of association, with the aim of judging the degree of *term-hood* of MWE candidates [38]. Such approaches that attempts to identify MWEs by means of statistical association measures are classified as Statistical Approaches. On the whole, statistical approaches are rarely stand-alone in MWE discovery, therefore, they are usually combined with some linguistic approaches.

Basically, there are several statistical measures that have been employed, and some of the commonly utilized ones include:

- Pointwise Mutual Information (pMI) measures the degree of cohesiveness/association of two words [40] with the following correlation:

$$pMI(x, y) = \frac{P(xy)}{P(x *)P(* y)}$$

- The correlation for measuring the mutual dependence (MD) between the two words [41] is:

$$MD(x, y) = \frac{P(xy)^2}{P(x *)P(* y)}$$

- Dice Coefficient computes the association measure (D) [42] with the following correlation:

$$D(x,y) = \frac{2f(xy)}{f(x) + f(y)}$$

Where:

** stands for any word*

P is probability

f is frequency of occurrence

Since these measures are based on frequency counts, low-frequency of a particular MWE in a corpus could affect the association measure. To avoid low-frequency estimates, Web hits have been deployed as alternative to mere corpus counts, considering the web is text-rich [6] [43].

Sometime in the last decade, a study was carried out to investigate the ranking of MWE candidates. 55 association measures (including Pointwise Mutual Information, t test, Log likelihood ratio, Phi, Pearson, Mutual dependency, Conditional probability, and so on) were combined via standard statistical-classification methods and modified to provide scores for ranking [41]. The author concluded that, generally, combined methods of ranking produced better results than individual Association Measure. He further observed that different Association Measure produced dissimilar results for different data, implying the absence of a suitable overall association measure. In other words, a specific Association Measure might produce a better result in one data but not for another. This inconsistency gave rise to the recommendation that not all 55 association measures be used in a combination method, also redundant and non-predictive ones should be eliminated in each case, while employing only properly fitted combination models [41].

2.3.1.3 Hybrid Linguistic-Statistical Approaches

Hybrid approaches combine both linguistic and statistical methods, taking advantage of the intrinsic value of each method in identifying MWEs. Usually, it is preceded by a syntactic pattern processing, then followed by either semantic analysis or statistical filters or both to improve the result. Several studies have utilized this method for Bulgarian [8], Slovene [44], and Polish [9]. They submitted that, though there were a

substantial number of false collocations extracted, it is relatively easy to separate them by manual inspection. Hence, their tool is useful as a semi-automatic extractor of MWEs.

Lately, a combination of syntactic, semantic and statistical methods has been employed [45]. Initially, MWEs candidates were extracted using syntactic rule-based method, then filtered with a combination of linguistic and statistic information to reject false ones. Rather than using a manually constructed syntactic structure/rule, the researchers developed a system which analysed several Serbian MWE e-dictionaries to extract the most frequent syntactic structures.

Therefore, in the current research, the hybrid linguistic-statistical approach would be exploited. This approach is selected because of its advantages over purely linguistic approach or purely statistical approach. The hybrid linguistic-statistical approach draws from the advantages of each approach while mitigating some of the inherent flaws in each. As a result, the Syntactic Rule-based extraction method would be used to extract candidate nominal MWEs from the corpora, afterwards, a statistical method would then be applied to rank the candidate nominal MWEs to eliminate false ones.

2.3.2 SpaCy: An Impressive Pre-processing Tool

Prior to the application of syntactic extraction method, Tokenization, POS tagging and possibly lemmatization, where necessary, has to be executed on the corpora. SpaCy, a Natural Language Processing package available in Python, is usually utilized for these pre-processing tasks [46], although NLTK is the common and most popular Natural Language Toolkit available in Python [47] [48]. In 2015, a study was carried out to compare various statistical dependency parsers available publicly [49] and spaCy was established to be the fastest greedy parser. Although relatively inaccurate, it falls within the top 2% of the best. Furthermore, the study attempted to systematically compare accuracy with speed and recommended spaCy and ClearNLPg for web-scale NLP, and Mate, RBG, Turbo, ClearNLP and Yara for highest accuracy NLP applications, such as dialog systems. Currently, ClearNLP works on Java but not Python and SpaCy has improved its accuracy in the 2.0 version [46].

Some preliminary test carried out with NLTK *pos_tag* method did not yield wonderful results. For example, POS tagging the sentence, [*The quick brown fox jumps over the*

lazy dog], rendered *brown* as a Noun (NN), instead of as an adjective (JJ). Also, POS tagging the sentence, [*I enjoy writing articles for BBC News. Checkout my other article by going to my profile section.*], rendered *checkout* as a Noun (NNP, proper noun, singular), instead of a verb. Of course, NLTK accuracy could be improved by carrying out further training before utilizing it [48]. Another widely accepted NLP tool available on Python is StanfordNLP, which was upgraded to Stanza, a standard way of using Stanford CoreNLP in Python [50] [51] [52]. Although StanfordNLP [53] or Stanza [54] is more accurate, the processing speed is very low to the point even the Stanford NLP Group recommend spaCy for faster Tokenization and Sentence Segmentation [55]. Accordingly, as a consequence of the aforementioned merits of SpaCy, it would be deployed in this research for the pre-processing tasks.

2.3.3 Evaluating MWE Discovery Process

As previously discussed, there is no single method perfected for discovery of MWE in corpora. Each method has its advantages and shortcomings. Though combining methods could benefit from the advantages of the constituent methods and mitigate some of their shortcomings, hybrid methods are relatively imperfect. Hence, it is imperative to evaluate the methods employed for accuracy and efficiency.

More often than not, method evaluation is simply reduced to the quality of the extracted MWEs. As important as the quality of the extracted information is, a method might simply be too tedious or unrealistic even if it delivers more accurate results. The system efficiency (in terms of human effort, computational demands in terms of hardware and software, overall running time, ability to be integrated into NLP tasks, amongst others) is equally important, otherwise the whole essence of extraction would be defeated if the system could barely be applied. As a result, both accuracy and efficiency measurements are important for evaluating a proposed system.

However, in terms of evaluating and verifying the validity of the underlying theoretical assumptions and linguistic theories, the accuracy of the extracted MWEs is more relevant than the efficiency of the system [38]. Furthermore, accuracy evaluation precedes efficiency evaluation, that is, the accuracy of the system has to be evaluated before performance. Certainly, the performance ability can be improved if the accuracy checks out, but a poor accuracy occasion impaired performance.

Despite decades of automatic MWE discovery efforts [6], there is no golden standard for evaluating accuracy or efficiency occasioned by the polemic nature and character of MWE itself. Nonetheless, various methods abound for evaluating MWE discovery, some of which include:

2.3.3.1 Posterior Human Judgements

Human linguistic experts were requested to select relevant MWEs from a list of the top n MWEs retrieved by a method. The proportion of the found positive entries was used as a measure of evaluation of the method [6]. Various statistical measures such as average precision rate, recall rate, precision at k , and so on could be employed. Definitely, if recall rate is to be determined correctly, the human expert would have to observe used corpora to determine if some MWEs were missed out by the discovery method.

Aside time consumption, another problem with this method of evaluation is its susceptibility to the expert's subjective interpretation of terms [38]. Besides, several experts could be employed and a consensus determined, even though it is not impeccable, it would show the dissensions.

2.3.3.2 Dictionaries or Other Standard Reference Lists/Lexicons

The extracted MWEs can be automatically evaluated by comparison with entries already available in these golden standard dictionaries or other specialized lexicons, especially if they are MWEs of special fields or specific domains [38]. This method presumes that the standard dictionary or reference list exists. More so, there is the assumption that entries unavailable in the dictionary or list are wrong [6].

2.3.3.3 Extrinsic Evaluation

As a consequence of the problematic nature of the aforementioned intrinsic evaluations, some researchers prefer extrinsic evaluation, that is, the discovery method can be evaluated on the basis on how the extracted MWEs improved the performance and accuracy of NLP applications that implement them. All things considered, the goal of MWE discovery is to improve NLP applications. Various studies have been executed to this effect with a Parsing NLP application [56] and a Machine Translation NLP application [57].

Certainly, there are advantages in this extrinsic evaluation, chief of which could be a step towards integrating performance evaluation with accuracy evaluation.

Furthermore, it could challenge researchers to develop MWE discovery tools that could easily be integrated into NLP applications rather than tools that stand aloof or do not greatly impact the NLP pipeline.

Nevertheless, one issue with this style of evaluation is its expansion towards integration into NLP applications [6], that is, despite producing good discovery, the integration is equally paramount to record a successful evaluation. In addition, this approach optimizes the evaluation problem and exposes the contentious evaluation methods employed for measuring the performance of NLP application. On balance, as regards time constraint in this research to develop NLP applications to test on, the human evaluators method would be employed for evaluation.

Chapter Three: Research Methodology

3.1 Introduction

This chapter summarizes the research methodology for this applied research on the automatic extraction and lexical encoding of Nominal MWEs from corpora. The research onion model [58], the standard for presenting research methods, was deployed in this work to discuss the Research Philosophy, Approach and Methodological Choice. Following that, the Procedures and Techniques to be employed were discussed.

3.2 Research Philosophy

The current work is an applied research whereby a new product was developed using basic research philosophy based on Positivism, which involves the use of existing scientific theory to develop hypotheses to be tested during the research process [59] [60] [61]. Furthermore, the objectives of designing and testing of the tool were based on existing valid scientific knowledge and methods in the subject area. Consequently, the objectives were tested and allow explanations that are measurable against acceptable knowledge, while the quantifiable results are subjected to statistical analysis [58].

Likewise, existing knowledge would guide the encoding process to ensure the re-integration of the discovered MWE into NLP. In any case, the positivist approach was adopted in investigating how MWEs are extracted from corpora and encoded for lexicons.

3.3 Research Approach

Subsequent to the research onion model, the deductive approach was employed in this work. This approach is particularly suited to the positivistic philosophical stance [61]. The research would be characterized by the development from general to particular, wherein the existing knowledge base was established which forms the basis of the research. Then, the specific knowledge obtained from the research process, in this case, the designed tool, was evaluated via data collection [62] [63]. In the end, the observations contributed to existing knowledge.

Deductive approach was used to arrive at how MWEs are extracted from corpora and encoded for lexicons. For example, if it is established that Nominal MWEs exhibit certain features, these features would be exploited in efforts to extract these nominal MWEs automatically from texts. Therefore, the deductive approach was pertinent in proffering answer to the following questions among others:

- 1) what are the features of MWEs that could distinguish them in corpora?
- 2) how best could these MWEs be extracted using these features?
- 3) which pre-existing tools could best assist in executing these in terms of programming languages, lexicons, corpora and other existing lexical tools?
- 4) what should constitute the form of the designed tool or solution?
- 5) what should constitute the format of the resultant MWE lexicon to be NLP application ready and aware?
- 6) how best can the designed tool or solution be evaluated to assess its efficiency and accuracy?

Certainly, following tool design guided by research findings, it was used to collect data (MWEs) in corpora and the results analysed and evaluated, using acceptable statistical methods to confirm, reject or improve the tool.

3.4 Quantitative Methodology

Quantitative research methods were also used in the process of data collection and analysis. The designed tool automatically extracted nominal MWE from texts, which were labelled by human evaluators as True or False. They equally listed non-extracted MWE from the text and their observations were used to the precision rate, the recall rate and the F1 score of the designed tool.

Furthermore, during the extraction stage, some statistical analysis, such as Pointwise Mutual Information (pMI), Mutual Dependence (MD) and Dice coefficient, were used to filter extracted Nominal MWEs.

3.5 Procedures and Techniques

Certain procedures and techniques have been selected in extracting and lexically-encoding Nominal MWEs. These procedures are in 4 stages with each employing

certain techniques. The stages are: Pre-processing, Extraction, Evaluation and Lexical Encoding.

3.5.1 Pre-processing

Prior to extracting MWEs from corpora, it is required to pre-process the corpora. Pre-processing involves tokenization, POS tagging and lemmatization of the words in the corpora. Therefore, spaCy was deployed for the pre-processing tasks. As mentioned in section 2.3.2, spaCy was selected for its balance between speed and accuracy. It would be used to execute the pre-processing tasks which becomes the input for the extraction stage.

3.5.2 Extraction

The hybrid linguistic-statistical approach was utilized for the extraction stage. Initially, the Syntactic Rule-based extraction method was used to extract candidate nominal MWEs from the pre-processed corpora. Then, a statistical method was applied to rank the candidate nominal MWEs to eliminate false ones.

The syntactic rule-based extraction module extracted Nominal MWE candidates using such rules as *Noun₁ Noun₂ [... Noun_n]*, *Noun₁ of Noun₂ [... Noun_n]*, and so on. Since Nominal MWEs are variant in their syntactic compositions, they could be made up of:

- more than one noun, either two or more (for example: bus stop, credit card, fire-fly, school bus, and so on),
- an adjective and a noun (for example: full moon, hot dog, sick leave, common room, free trade, and so on),
- a verb and a noun (for example: washing machine, swimming pool, driving licence, drinking water, helping hand, stop clock, post office, taste buds, search engine, and so on),
- a noun and a verb (for example: sunrise, breastfeeding, rainfall, horse riding, Air conditioning, House warming, Bird watching, phone call, house arrest, stone's throw, and so on),
- an adjective and a verb (for example: dry-cleaning, public speaking, speed walking, time taken, and so on),
- a verb and a preposition/adverb (for example: checkout, check-up, take-off, setup, draw back, and so on),
- a noun and a prepositional phrase (for example: brother-in-law, and so on),

- a preposition/adverb and a noun (for example: underworld, past lives, onlooker, bystander, in service, and so on),
- a noun and an adjective (for example: Lifelong, home sick, blood red, handpicked, price worthy, trustworthy, and so on)
- nouns interjected with prepositions such as: of, for (for example: Duke of London, and so on)

After extracting the candidate Nominal MWEs, they were ranked using a couple of Association Measures (AM) to eliminate false MWEs, since not all word combinations following the above syntactic rules are MWEs. For example, even though *hot dog* is an MWE, *hot food* is not, though they are both composed of an adjective and a noun. Various AMs were used to eliminate false MWEs, and decisions were made from consideration of Pointwise Mutual Information (pMI), Mutual Dependence (MD) and Dice coefficient.

Collectively, in computing these AMs, an important component is the frequency of the co-occurrence of the words of the extracted MWEs and of the individual component words in a corpus. Thus, a corpus is required to extract this component. Since the target language is only English, the 100-million words British National Corpus (BNC) [64] was employed. This was selected because it is free, vast and contains words from a wide range of genres (for example fiction, magazines, newspapers, and academic) [65].

3.5.3 Evaluation

The extracted MWE lists were presented to two human evaluators to label as relevant or irrelevant. They could also indicate Nominal MWEs that were missed out by the extraction process. Their observations were used to compute the precision and recall rate of the extraction tool, as well as its F_1 score, which were relevant in confirming, rejecting or making recommendations for improving the tool.

3.5.4 Lexical Encoding

The Nominal MWEs that have been validated by the human evaluators were lexically encoded. A semi-flexible encoding was employed with its three core elements and two optional components described in section 2.2.2. This encoding used UD relations

for denoting relationship between words and UD annotations for parts of speech, and so on.

3.6 Conclusion

In this applied research on the automatic extraction and lexical encoding of Nominal MWEs from corpora, a positivist philosophical stance was used; existing valid scientific knowledge and methods accepted as the basis of the research. Thus, employing a deductive approach, specific knowledge was derived from general scientific theory and evaluated. Quantitative (statistical) methods were equally employed for the data analysis. The whole procedure was executed in four stages, with various techniques employed in each encompassing the Pre-processing, Extraction, Evaluation and Lexical Encoding Stages. Finally, a tool was designed to deploy these stages, which methodology is discussed in the next chapter.

Chapter Four: Design and Implementation

4.1 Introduction

The crux of this applied research is to design a tool that would automatically extract and lexically encode Nominal MWEs from corpora. In the preceding chapter, the techniques/procedures have been enumerated. The current chapter basically entails the design and implementation of the tool. Since software development is required in this research, one of the existing Software Development Life Cycle (SDLC) models was utilized.

4.2 Waterfall SDLC Model

A Software Development Life Cycle (SDLC) model offers a systematic process for building and delivering software applications [66]. Generally, an SDLC has the following stages: Planning and requirement analysis, Designing project architecture, Development and programming, Testing, and Deployment [67]. After careful analysis and thorough comparison of the following SDLC models; Agile SDLC, DevOps SDLC, Iterative SDLC, Lean SDLC, Spiral SDLC, Waterfall SDLC and V Model as well as the Rapid Application Development approach, the Waterfall SDLC was selected owing to its appropriateness for small to middle sized projects, where requirements are clear and the product definition as well as the technologies stack is stable [68] [69]. Characteristically, the Waterfall model follows the SDLC stages sequentially.

4.2.1 Requirement specification and analysis

As has been established so far in this work, the broad requirements of the tool are to extract Nominal MWEs from corpora and lexically encode the MWEs that human evaluators have validated. Since the human evaluators or linguistic experts could be locally-dispersed, it is paramount that the tool has a web interface.

Furthermore, the tool was able to potentially improve its extraction capacity by learning and correcting itself. For instance, if a particular MWE candidate previously dropped by the tool was adopted by both human evaluators, the tool would store such candidate to subsequently present it among the filtered MWEs. On the other hand, if the tool accepted a candidate MWE that the two human evaluators later rejected, the tool would learn that MWE and not present it in the future among

filtered MWEs. This requirement is very crucial as it makes the tool self-learning and intelligent.

4.2.2 Designing project architecture: A modular architectural design

For the software architecture, a modular approach was adopted which allows for easier software evaluation, since respective parts can be isolated and evaluated leading to the ensuing parts. Four modules were designed corresponding to the stages discussed in section 3.5. Also, an additional module was included to execute the optional conversion of the lexical encoding to an XML-encoded lexicon, that conforms to the LMF standard, which is the ISO standard for Natural Language Processing (NLP) lexicons [70] [34] [35] [71]. Figure 1 shows how the modules fit together:

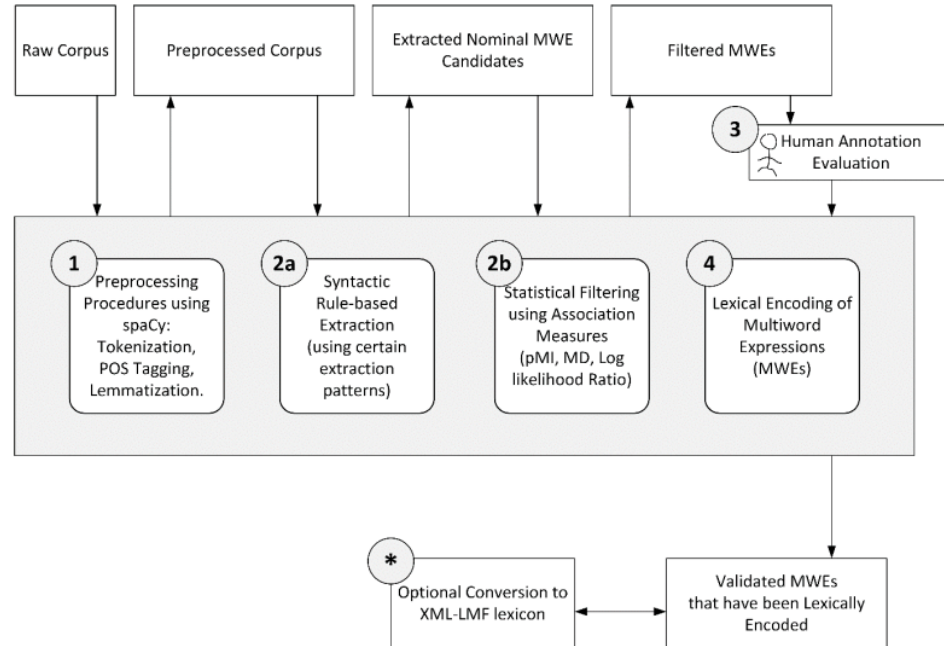


Figure 1: Modules of the Design Architecture

4.3 Design

The design of the tool followed the modular architecture. The user has the option to abort at any of the modules, if he/she is only interested in the result of a particular modules. The flowchart of the extraction is displayed in Figures 2 and 3.

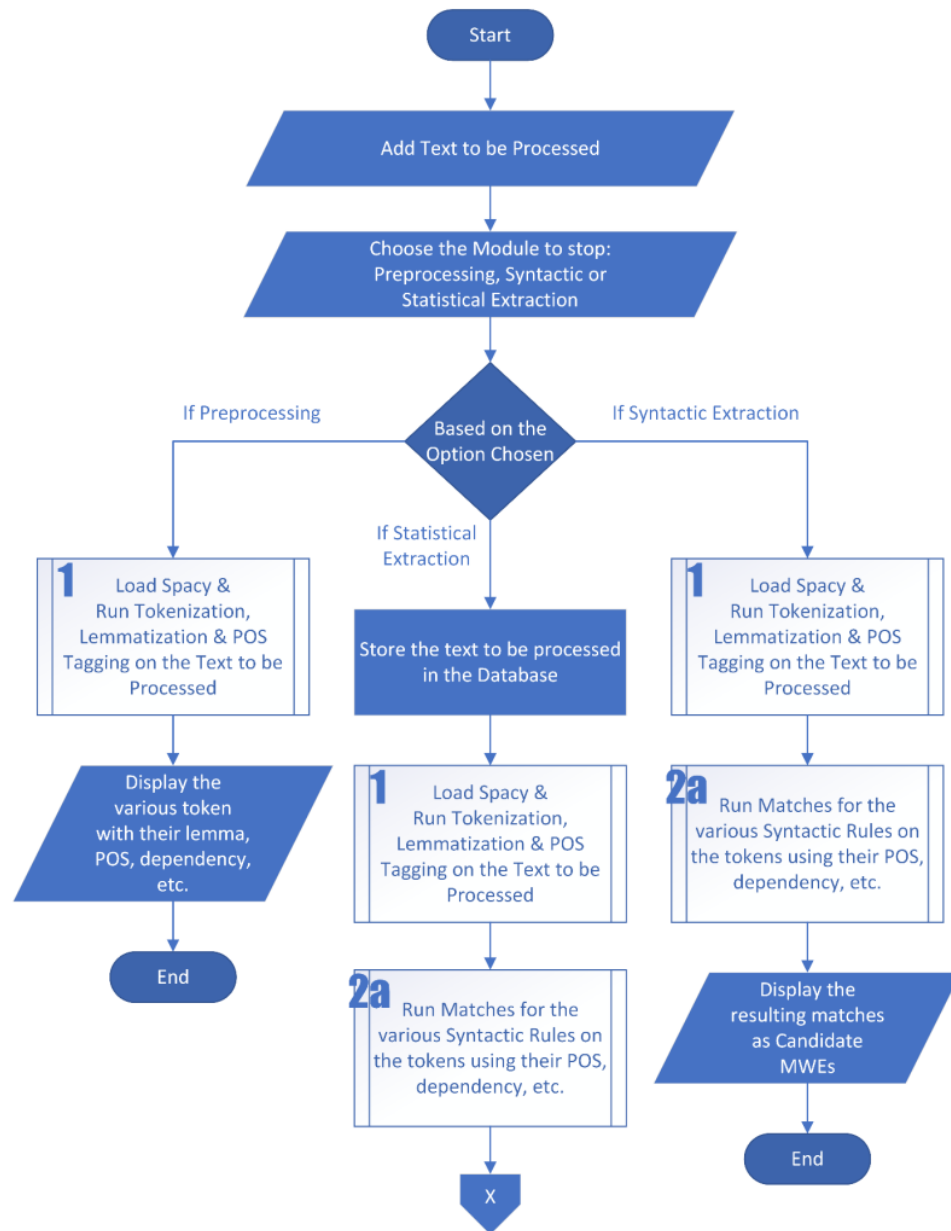


Figure 2: Flowchart of the Extraction Section

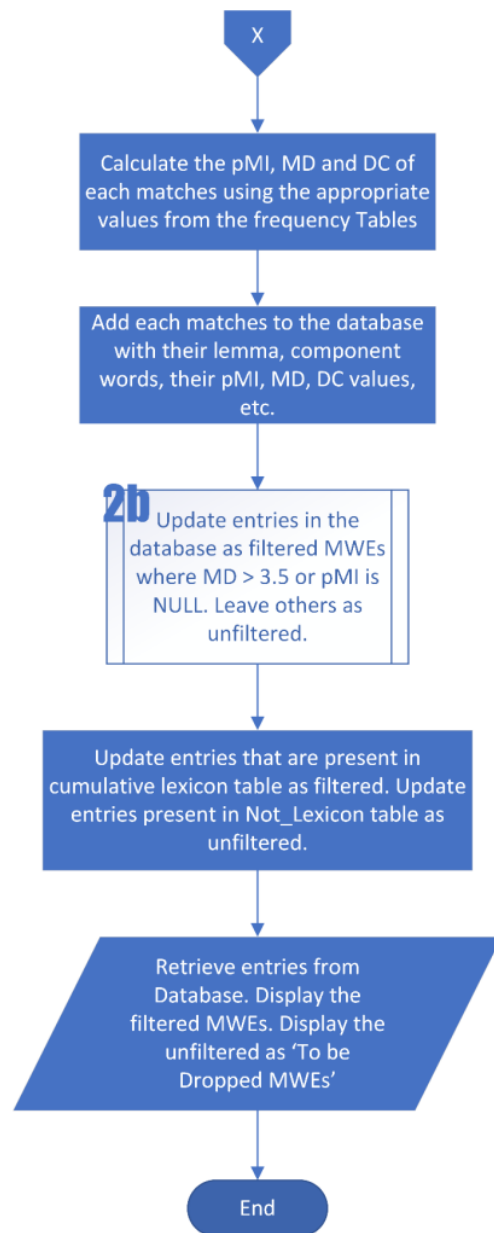


Figure 3: Flowchart of the Extraction section (continuation)

The pre-processing and syntactical extraction options simply display results for on-screen viewing, while nothing is deposited on the database. However, the statistical extraction option yields results to the database that the human evaluators can

evaluate. Therefore, only after statistical extraction would human evaluation be carried out. The flowchart for the human evaluation stage is displayed in Figure 4.

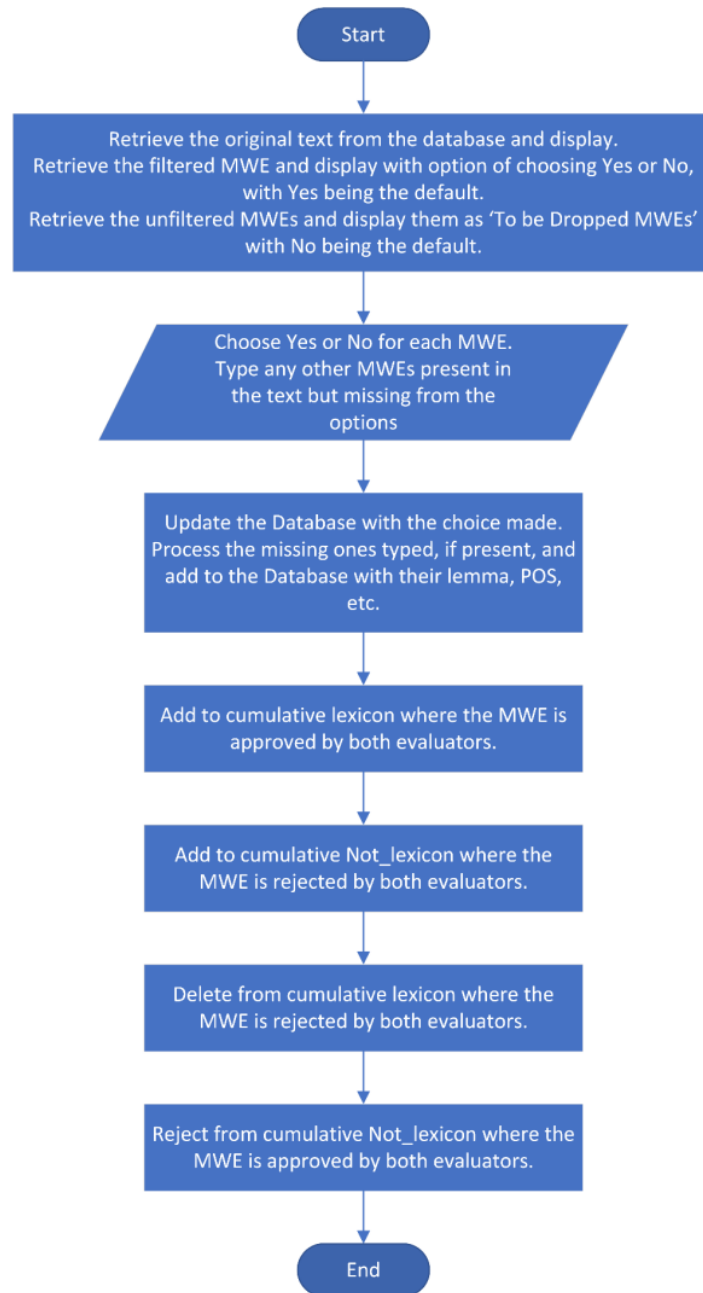


Figure 4: Flowchart for MWE Evaluation

4.4 Implementation

The tool was implemented with a user-friendly interface to allow for easy utilization and engagement. Specifically, a web interface was employed. As summarized in Figure 5, this web interface is backed by PHP and interacts with the user by relaying input to either Python for the text analysis or to the MySQL database for storage as the case might be. Subsequently, Python as the engine house of the entire tool execute the text analysis and reports either to the MySQL Database for storage or to the browser interface via PHP for display as the case might be.

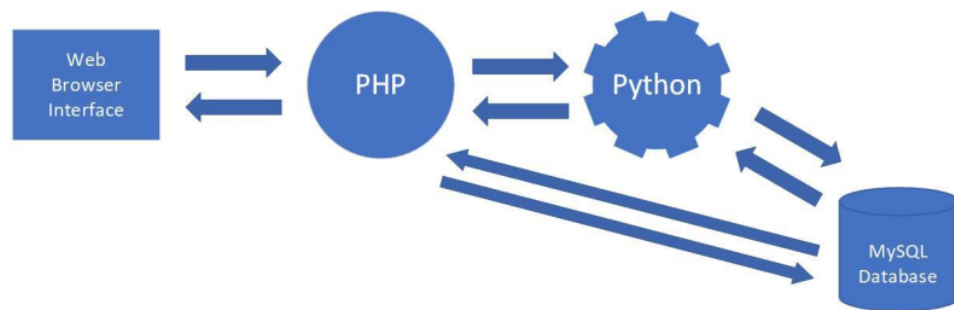
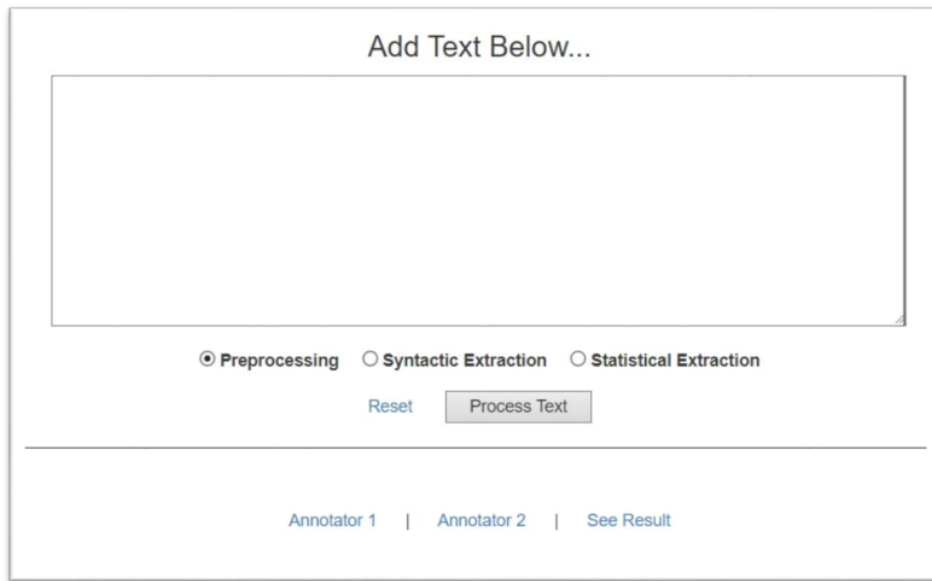


Figure 5: A Skeletal Presentation of the Workflow

The web interface displayed in Figure 6 was designed to provide a step by step interface to the entire working of the tool. It provides the option to simply run and stop at Pre-processing or Syntactic Extraction or run all the way to Statistical Extraction, and it is assumed to users an insight into the working of the tool. Furthermore, users might just be interested in the result of a particular module.



The image shows a web interface for a text processing tool. At the top, there is a header "Add Text Below..." above a large, empty rectangular text input area. Below the input area, there are three radio button options: "Preprocessing" (which is selected), "Syntactic Extraction", and "Statistical Extraction". Under these options are two buttons: a blue "Reset" button and a grey "Process Text" button. At the bottom of the interface, there is a horizontal line, and below it, three links: "Annotator 1", "Annotator 2", and "See Result", separated by vertical bars.

Figure 6: Web Interface of the Tool

Depending on the option selected at the time the user clicks on [Process Text] button, different feedbacks would be received. The pre-processing option would simply provide the various token in the text and their POS, lemma, etc as shown in Figure 4. At the core, the pre-processing option runs the `[pos100.py]` script detailed in the appendix 2. It processes the text submitted by PHP using the spaCy module in Python to execute the tokenization, lemmatization, POS tagging, etc of the text; which result is subsequently sent back via PHP to the browser display.

Preprocessing								
Madrid and nearby cities will see restrictions enforced by 7,000 police. So says social media. I ate hot dog yesterday. The capital has been at the centre of a political row, with the centre-right city authorities challenging the Socialist-led government's demands. Cases are down and a state of emergency is unjustified, say city officials. Madrid health minister Enrique Ruiz Escudero insisted that measures already in place were working and that the national government order was "a measure no Madrileño will understand".								
Go back to Home								
S/N	TOKEN	POS	TAG	LEMMA	DESCRIPTION	DEPENDENCY	HEAD WORD	DESCRIPTION
1	Madrid	PROPN	NNP	Madrid	noun, proper singular	nsubj	see	nominal subject
2	nearby	ADJ	JJ	nearby	adjective	amod	cities	adjectival modifier
3	cities	NOUN	NNS	city	noun, plural	conj	Madrid	conjunct
4	restrictions	NOUN	NNS	restriction	noun, plural	doobj	see	direct object
5	enforced	VERB	VBN	enforce	verb, past participle	acl	restrictions	clausal modifier of noun (adjectival clause)

Figure 7: Feedback Interface for Pre-Processing

The Syntactic Extraction option gives the result of the syntactic rule-based extraction that extracts Nominal MWEs according to certain pre-defined syntactic rules. Its interface is as shown in Figure 5. At its core, the syntactic extraction option runs the `[pos200.py]` script detailed in the appendix 3. In addition to executing pre-processing (tokenization, lemmatization, POS tagging, etc) like the `[pos100.py]` script, this script runs some pre-defined syntactic rules on the pre-processed text to extract multiword expressions that match these rules as discussed in section 3.5.2. The matched Nominal MWEs are sent back via PHP to the browser display.

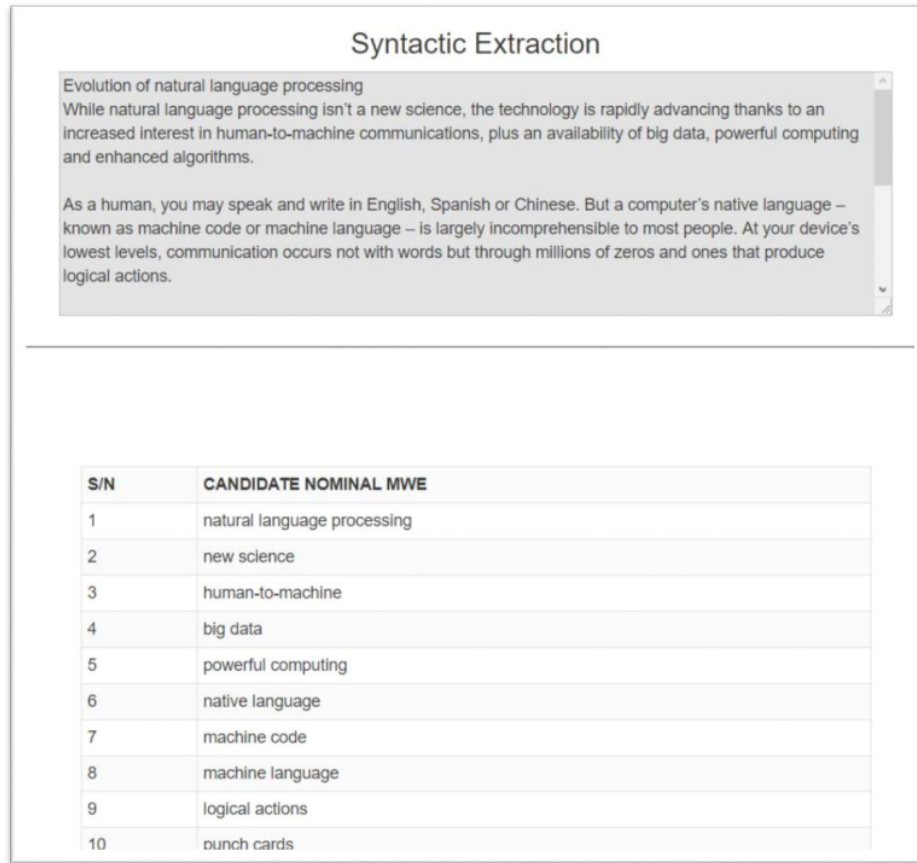


Figure 8: Syntactic Extraction Feedback Interface

The Statistical Extraction option went beyond just extracting on the basis of syntactic rules; it filters the extracted MWEs on the basis of various AMs results. Its feedback interface provides filtered MWEs as well as others that do not meet the threshold. The interface is displayed in Figure 9. At its core runs the *pos300.py* script detailed in the appendix 4. Aside executing pre-processing and syntactic extraction like the *pos200.py* script, the *pos300.py* dissect each extracted MWEs and retrieves their frequencies from the frequency table. Utilizing these frequencies, the script computes the pMI, MD and DC of the MWEs. The extracted MWEs with their lemma, POS, component words, pMI, MD, DC etc were inserted into the database. On the basis of the association measures, the extracted MWEs were classified as filtered or non-

filtered. These were further classified using entries in the cumulative lexicon or those that have been rejected in the past. Finally, the records from the database were retrieved and display on the browser. It is noteworthy that the result of any previous statistical evaluation was deleted from the database prior to saving another evaluation in the same database to ensure that the human evaluators were only dealing with the current text.

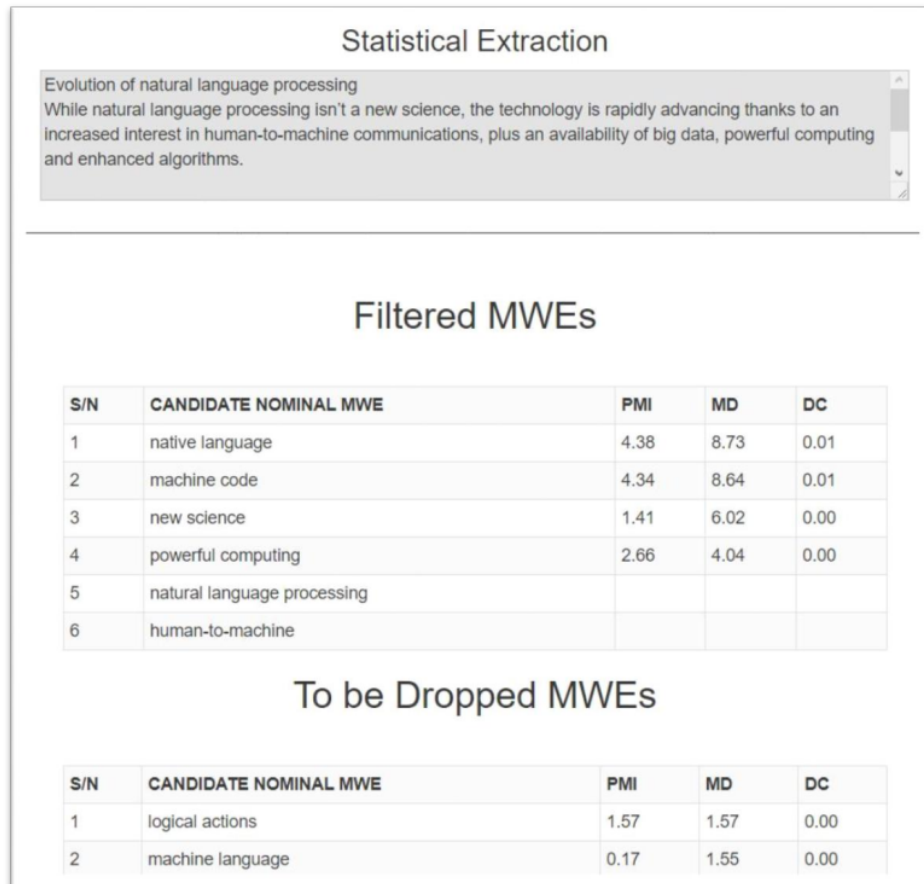


Figure 9: Statistical Extraction Feedback Interface

It is only after a statistical extraction has been executed on a text before human evaluation is possible. The human evaluation interface retrieves the result of the previous statistical extraction from the database and presented with the option of choosing yes or no for each MWE as displayed in Figure 10. First, the original text from

which the extraction was done is presented for the perusal of the evaluator. Secondly, the filtered MWEs were presented with options of [Yes] or [No], with yes as the default. Thirdly, the MWEs that the tool proposed to drop were presented with the options of [Yes] or [No] with no as the default. It is pertinent to clarify here that these options are for the evaluator to rate the particular MWE. Finally, a text area was presented for the evaluator to type any missing MWEs that were present in the original text, but not extracted by the tool.

Original Text

Hot dog is an essential movement.

MWEs Retained

S/N	NOMINAL MWE	YOUR DECISION: Is it an MWE, Yes or No
1	Hot dog	☒ Yes ☐ No

MWEs Dropped

S/N	NOMINAL MWE	YOUR DECISION: Is it an MWE, Yes or No
1	essential movement	☐ Yes ☒ No

Please, type below any Nominal MWEs that are missing above.

Separate your entries with commas. E.g. Hot dog, fast food, free trade

Note that the typed MWEs must be present in original text. Any entries not present in the text would be disregarded.

Figure 10: Human Evaluation Interface

When the evaluator clicks Done, his/her input are uploaded to the database appropriately by the `[processannotate.php]` script as detailed in the appendix 14, and a feedback is displayed on the browser.

Accordingly, when the `[See Result]` link is click on the home page, one is able to view the result of the current extraction, based on the human evaluations. There are two buttons at the bottom of the page: `[Process XML-LMF for Current Text]` and `[Process`

Cumulative XML-LMF]. The function of these buttons is to generate the lexical encoding of the Validated MWEs in the XML-LMF format. The [*Process XML-LMF for Current Text*] button generates only the current validated MWEs stored in the [*mwes*] table, while the [*Process Cumulative XML-LMF*] button generates the cumulative extracted MWEs derived from the [*lexicon*] table.

4.4.1 Generating Frequency Table

As previously discussed in section 2.3.1.2, the pMI, MD and DC values of the MWEs are computed using the frequencies of the component words and the MWE in a corpus. The corpus selected to achieve this is the BNC. Therefore, in a bid to attain these frequencies, the BNC would have to be loaded and converted to text format since it was originally in XML format prior to obtaining the word frequencies. These processes are economically expensive, hence to save time and resources, the BNC was converted to text and the frequencies of the various distinct words were computed and stored in a frequency table. This table was subsequently loaded to assess the frequencies of words.

The BNC was downloaded from <https://ota.bodleian.ox.ac.uk/repository/xmlui/handle/20.500.12024/2554>. It is the XML edition and 538.34 MB in size [72]. It is composed of various XML scattered across several folders. A preview of a portion of the XML files is shown in Figure 11.

```

<u who="D94PS000">
<s n="4">
  <w c5="AV0" hw="well" pos="ADV">Well </w>
  <w c5="PNP" hw="we" pos="PRON">we</w>
  <w c5="VM0" hw="will" pos="VERB">'ll </w>
  <w c5="VDI" hw="do" pos="VERB">do </w>
  <w c5="DT0" hw="that" pos="ADJ">that</w>
  <c c5="PUN">, </c>
  <w c5="PNP" hw="i" pos="PRON">I </w>
  <w c5="VVB" hw="see" pos="VERB">see </w>
  <w c5="DTQ" hw="what" pos="PRON">what </w>
  <w c5="VVB" hw="see" pos="VERB">see </w>
  <w c5="VZ" hw="say" pos="VERB">says </w>
  <w c5="DT0" hw="any" pos="ADJ">any </w>
  <w c5="NN1" hw="rate" pos="SUBST">rate </w>
  <w c5="PRP" hw="on" pos="PREP">on </w>
  <w c5="NP0" hw="monday" pos="SUBST">Monday</w>
  <c c5="PUN">.</c>
</s>
</u>
<u who="D94PSUNK">
  <s n="5">
    <w c5="CJC" hw="but" pos="CONJ">But </w>
    <w c5="PNP" hw="i" pos="PRON">I </w>
    <w c5="VDB" hw="do" pos="VERB">do</w>
    <w c5="XX0" hw="not" pos="ADV">n't </w>
    <w c5="VVI" hw="think" pos="VERB">think </w>
    <w c5="PNP" hw="we" pos="PRON">we </w>
    <w c5="VM0" hw="should" pos="VERB">should </w>
    <w c5="VVI" hw="give" pos="VERB">give </w>
  </s>
</u>

```

Figure 11: A snippet of an XML file in the BNC

In python, with the code in Figure 12, the texts in the various xml files were extracted and saved as a text file: BNC.txt.

```

import glob
import xml.etree.ElementTree as ET

filenames = glob.glob("C:\\Temp\\2554\\download\\Texts\\**\\*.xml", recursive = True)

out_lines = []
for filename in filenames:
    with open(filename, 'r', encoding="utf-8") as content:
        tree = ET.parse(content)
        root = tree.getroot()
        for w in root.iter('s'):

            for child in w:
                feli = child.tag
                if feli == 'w':
                    lemma = child.get('hw')
                    pos = child.get('pos')
                    tag = child.get('c5')
                    out_lines.append(child.text)
                if feli == 'c':
                    out_lines.append(child.text)
                    tag = child.get('c5')

            out_lines.append("\n")

with open("C:\\Temp\\bnc.txt", "w", encoding="utf-8") as out_file:
    for line in out_lines:
        out_file.write(str(line))

```

Figure 12: Code to extract text from BNC XML files

The resultant text file, BNC.txt, is 539 MB. This file was processed to generate the various distinct words with their respective frequency. The tokenization function was executed with spaCy, while the *Counter* function from the 'collections' module in python was used to extract the frequency table (the distinct words with their frequencies). The frequency table was subsequently saved in a file named BNC_Frequency.txt to be called up whenever the frequency of a word is required. This process was executed with the code in Figure 13.

```

from collections import Counter
import pickle
import spacy
nlp = spacy.load("en_core_web_sm", disable=["tagger", "parser", "ner"])

nlp.Defaults.stop_words.add("=")
nlp.Defaults.stop_words.add("\n\n")
nlp.Defaults.stop_words.add("\n ")

nlp.max_length = 600000000

file_name = 'C:\\Temp\\bnc.txt'
f = open(file_name, 'r', encoding="utf-8")
file_text = f.read()

conten = nlp(file_text)
words = [token.text.lower() for token in conten if not token.is_stop and not token.is_punct and token.text.isalnum()]

f.close()
del file_text

word_freq = Counter(words)

lily = len(words)
counter2 = Counter({'DTotalWords2020': lily })

dword = word_freq + counter2

the_filename = 'C:\\Temp\\BNC_Frequency.txt'
with open(the_filename, 'wb') as fa:
    pickle.dump(dword, fa)

print('Done!')

```

Figure 13: Code to extract and save frequency table from BNC text

The resulting file, BNC_Frequency.txt, contains only single words and their frequencies. However, bigrams' frequencies were also necessary for the computation of pMI, MD, etc. The integration of bigrams frequency in the BNC was executed with the code displayed in Figure 14 and saved for subsequent use.

```

from nltk import bigrams
from collections import Counter
import pickle
import spacy
nlp = spacy.load("en_core_web_sm", disable=["tagger", "parser", "ner"])

nlp.Defaults.stop_words.add("=")
nlp.Defaults.stop_words.add("\n\n")
nlp.Defaults.stop_words.add("\n ")

nlp.max_length = 600000000

file_name = 'C:\\Temp\\bnc.txt'
f = open(file_name, 'r', encoding="utf-8")
file_text = f.read()

conten = nlp(file_text)
words = [token.text.lower() for token in conten if not token.is_stop and not token.is_punct and token.text.isalnum()]

f.close()
del file_text

dw = list(bigrams(words))
wd = [' '.join(c) for c in dw]

word_freq = Counter(wd)

lily = len(wd)
counter2 = Counter({'DTotalWords1919': lily })
dword = word_freq + counter2

the_filename = 'C:\\Temp\\BNC_Frequency_BiGram.txt'
with open(the_filename, 'wb') as fa:
    pickle.dump(dword, fa)

print('Done!')

```

Figure 14: Code to extract and save frequency of Bigrams

Admittedly, the importation of bigram function of the *nltk* module in python and subsequent generation of bigrams in the BNC text differentiate the codes in Figures 13 and 12. These bigrams frequencies obtained using the *Counter* function of the 'collections' module were saved as [*BNC_Frequency_BiGram.txt*]. Thus, the files [*BNC_Frequency.txt*] and [*BNC_Frequency_BiGram.txt*] are critical files that were loaded in the [*pos300.py*] for statistical extraction. The two files were subsequently loaded in order to obtain frequencies of words and bigrams for computation of pMI, MD and DC.

4.4.2 Database and Tables

As previously reiterated, MySQL was utilized for the database and the SQL dump of the full database is detailed in the appendix 1. Some of the important table of the database are the *mwes*, *lexicon* and *not_lexicon* table. The *mwes* is the location where the extracted MWEs from statistical extraction are stored with 13 fields as displayed in Figure 15. The field *mwe* contains the MWE, *pmi* contains the pMI value of the MWE, *md* contains the MD value, *dc* contains the DC value, *archived* denotes the options of dropping or retaining the MWE, *choice1* and *choice2* contains the decision of the first and second evaluators respectively. The remaining 5 fields contain entries for the lexical encoding of the MWE. The field *mwedes* contains the MWE description, *compdes* contains the Component Description, *combpatt* contains the Combination Pattern, *mwemeaning* contains the meaning of the MWE (if it can be obtained from wordnet) and *mwesample* contains examples of sentences where the MWE is used (if it can be obtained from wordnet). The *mwes* table is emptied every time there is a new statistical extraction to be done.

#	Name	Type	Collation	Attributes	Null	Default
1	sn 	int(11)			No	None
2	mwe	varchar(250)	latin1_swedish_ci		Yes	NULL
3	pmi	decimal(10,2)			Yes	NULL
4	md	decimal(10,2)			Yes	NULL
5	dc	decimal(10,2)			Yes	NULL
6	archived	varchar(3)	latin1_swedish_ci		No	Yes
7	choice1	varchar(3)	latin1_swedish_ci		No	No
8	choice2	varchar(3)	latin1_swedish_ci		No	No
9	mwedes	varchar(250)	latin1_swedish_ci		Yes	NULL
10	compdes	varchar(250)	latin1_swedish_ci		Yes	NULL
11	combpas	varchar(250)	latin1_swedish_ci		Yes	NULL
12	mwemeaning	text	latin1_swedish_ci		Yes	NULL
13	mwesample	text	latin1_swedish_ci		Yes	NULL

Figure 15: Description of the *mwes* Table

The *lexicon* table holds cumulatively the MWEs that have been accepted previously by both evaluators and is continuously updated with every new MWEs accepted by them. Likewise, any entry rejected by both evaluators during text evaluation is deleted from the table. Conversely, the *not_lexicon* table holds cumulatively the MWEs that have been rejected previously by both evaluators. Entries in this table are also deleted if they are accepted as MWE by both evaluators during text evaluation session. These two tables form the memory of the tool used to improve the judgement of whether a term is MWE or not, in addition to the values of the AMs. They are very similar in structure and contain the same fields: *sn*, *mwe*, *archived*, *mwedes*, *compdes*, *combpas*, *mwemeaning* and *mwesample*. These are simply obtained from the *mwes* table at the point of entry into the table, whether *lexicon* or *not_lexicon*.

4.4.3 Setting Up Python

Python 3.7 was selected for this project, even though versions 2.7 and 3.5 upwards would work because the spaCy package that was employed only works with python 2.7 or 3.5 upwards. Also the system requirement is 64-bit [73].

SpaCy, mysql.connector and nltk were installed on python for this project. SpaCy was installed with PIP using the following code on command prompt: *pip install spacy*. Additional data tables for lemmatization were installed using the command: *pip install spacy_lookups_data* [73]. Also, the English language pretrained model to identify POS tags, syntactic dependencies, etc was installed using the code: *python -m spacy download en_core_web_sm*.

The mysql.connector package was useful to enable Python connect to MySQL database. This package was installed using the code: *pip install mysql.connector*. Finally, the nltk package was installed to enable the tool lookup meanings of words as well as sample sentences in the lexical encoding phase. It was installed using the command: *pip install nltk*. Following which, the wordnet lexical database, a part of the nltk corpus, was downloaded using the command: *nltk.download('wordnet')*.

Chapter Five: Analysis and Evaluation

5.1 Introduction

The tool, which is available at <https://vps64990.inmotionhosting.com/~nislgt5/mwe>, was evaluated for accuracy by requesting two human evaluators to label extracted MWEs as correct or incorrect, which was used to compute the precision and recall rate, and the F_1 score. The interface for the human evaluation is web-based and the link was sent to the two human evaluators. The evaluation page presents all the extracted MWEs with an option for each evaluator to approve or reject each candidate as MWEs. Each evaluator's decision was then updated to the database.

5.2 Results and Discussion

Various texts were processed and the resulting extracted MWEs evaluated by the two human evaluators. The precision rate usually ranged from 0.39 to 0.75, while the recall rate ranged from 0.64 to 0.96.

The low scores of the rates were usually recorded when there are new MWEs like social media, big data, deep learning, machine learning, and so on. The tool would extract these but after the statistical filtering, using the association measures of pMI, MD and DC, would classify them as MWEs to be dropped, because of low scores according to AMs. These low scores are due to the low frequencies or even total absence of the new MWEs in BNC which was employed for computing word frequencies. It is imperative to state here that these new MWEs became popular around the 2000s. The BNC did not contain much of these new MWEs, though it is 100-million words and drawn from a wide range of sources and genres. The corpus was actually done in the late 20th century to be a representative sample of British English at the time. Since language is always evolving, a base corpus is required that is always expanding and representative of the language at any given time.

Currently, there is the *web* corpus that is comprised of 14 billion words derived from over 95,000 websites [74]. However, this corpus is not free. Also, the size of the corpus renders it economically intensive beside the fact that terms coined post 2017 would not be captured. Therefore, the situation narrows down to a self-evolving corpus, which would degenerate to an exponentially increasing file size unless archaic words are deleted hence limiting the applicability of the corpus.

Another limitation with the *iweb* corpus compared to the BNC is the former was compiled from the web, while the latter was compiled from a balanced range of sources: spoken, fiction, newspapers, magazines, and academic. It could be argued that the web is all-encompassing but the context and style of the essays are paramount. There is a slight variation between a newspaper version of a news outlet and its web version. Although debatable, the point remains that the BNC is a balanced effort from several genre while the *iweb* is solely sourced from the web even though it contains various websites.

Owing to the issue of limiting corpus frequency table, some researchers have suggested searching the web for frequencies of words and terms [43] [6]. Keller & Lapata, 2013 carried out such searches in comparison to corpus searches. However, their web searches were done using a simple heuristic based on queries to the search engines AltaVista and Google and adding up the number of pages that reported matches. Such heuristic methods could not be used in a tool that requires automatic evaluation to filter MWEs. Besides, the computations of pMI and MD requires probability aside frequency of occurrence. This probability becomes impossible to define as the total number of words on the web is indeterminable. Conclusively, the idea of using the live web for word frequencies is unfeasible.

Furthermore, the limitation of word frequencies from corpus, among other issues, have encouraged some scholars to explore pure semantic methods, rather than the statistical methods or hybrid extraction methods that include statistical evaluation. They used semantic interpretations, context analysis or translations comparison [36] [39], using either supervised machine learning on annotated data or comparing two or more parallel translations of the same text. One of the impediments encountered was that they depend on annotated data or parallel translations which are not readily available or relatively small and specific. Therefore, these methods may not be applicable to highly ambiguous MWEs [6]. On the other hand, Syntactic-Statistical methods do not rely on annotated data or parallel translations.

Occasionally, low figures of precision and recall rates are not promoted by new MWEs. In other words, there are instances where low precision and recall rates are reported but new MWEs are not recorded in the text. This is a stark reminder that Association Measures are not the absolute determinant of MWEs and human evaluation is *sine qua non*. It is noteworthy to state here that AMs only measure collocation strength

which is an important feature of MWE among others. Indeed, there are some collocations that are not MWEs. MWEs exhibit other features such as non-compositionality, ambiguity, and so on as discussed in section 2.3 and AMs do not measure them. Though semantic methods have been attempted to measure non-compositionality, the limitations of such methods have been mentioned in the preceding paragraph. Therefore, human evaluations are key to MWE discovery because MWEs formation can be quite arbitrary and do not follow set rules or logic at all times.

In spite of obtaining word frequency from corpora, AMs are necessary for any MWE extraction using the syntactic method. In reality, any form of AMs from any corpora is appreciated than null because without them, all collocations matching the specified syntactic rules would be extracted and inundate the human evaluators. Hence, any reasonable form of AMs would unburden the human evaluation and facilitate their efforts.

Importantly, the self-improving feature of the designed tool ensures reduction of the work load of human evaluation. If a term has been approved by both evaluators, such term is subsequently approved by the tool even if its AM values are lower than the acceptable threshold. Conversely, if a term has been rejected by both evaluators, such term is subsequently rejected by the tool even if its AM values are higher than the threshold. With this self-improving feature, the designed tool would improve with time as expert linguistics evaluate MWEs processed by it.

5.3 Conclusion

The evaluation of the designed tool was carried out by human evaluation of the extracted MWEs, which yielded a range of 0.39 to 0.75 for precision rate and 0.64 to 0.96 for recall rate. Though the use of BNC as the corpus of the calculation for AMs played a role in these figures, it is not entirely responsible for the occasional low figures. Association Measures only measure collocation strength, as such they cannot correctly determine MWEs at all times. Collocation is only one of the many other features of MWEs, though it is an essential feature. The other features of MWEs are not easily captured and evaluated quantitatively. Even attempts that used such features did not yield a perfect result [6] [39] [36]. Human evaluation, therefore, remains a *sine que non* for automatic MWE discovery from texts. The designed tool

and the techniques employed in this project are, nonetheless, useful in that it ensures that human evaluators are not inundated with too much false MWEs. Also, the designed tool would help in the process of MWE lexicon formation by generating XML-LMF lexicon of validated MWEs. The MWEs are lexical encoded using the same lexical information generated during the pre-processing stage and used for the syntactic extraction.

Chapter Six: Conclusions & Recommendations

6.1 Summary

NLP assists computers to interpret and manipulate human language. It is important because it fosters communication, knowledge management and discovery. Some applications of NLP are Machine Translation, Information Retrieval especially web search, etc.

MWEs are part of what NLP has to deal with, and they are replete in our everyday speech and writings. These MWEs are problematic because even though they are multi-words, they often do not follow grammar rules unlike other group of words like phrases, clauses, etc. They sometimes behave like a single word, exhibiting lexical, syntactic, semantic, pragmatic and/or statistical idiomaticity or peculiarities. To be clear, MWEs are not homogeneous and can occur in a wide range of lexical and syntactic configurations, for example, nominal, conjunctions, prepositions, Idioms, Verb-Particle Construction (VPC), Light-Verb Construction (LVC), among others.

Therefore, MWEs have to be appropriately identified and managed in texts if NLP applications are to be accurate. Ignoring them in NLP tasks generate ungrammatical, ambiguous or unnatural output leading to probable information loss. In the past, MWEs discovery in texts has been the focus instead of the integration of automatically acquired or manually compiled MWE resources in applications. These discovery efforts would be better appreciated if the extracted MWEs are linked to lexicons or put in lexicon-format for future use by NLP applications. This study is an attempt to use existing knowledge and methods to extract Nominal MWEs from texts, validate them using human evaluation and put such validated MWEs into lexicon for future use by NLP applications. In lexically encoding the MWEs, a standard MWE lexicon format was employed so that NLP applications can easily be adapted to implement it. Since NLP applications vary in their use of lexicon, the study would include a conversion of such lexicon into XML-LMF format, which is the ISO standard for NLP lexicons.

The tool was designed using a positivist philosophical stance, with a deductive approach and quantitative methodology, with the Waterfall SDLC Model. Precisely, the tool featured a web-based interface for easy interaction and enable linguistic experts who might be dispersed spatially to access it. Furthermore, it was able to improve its extraction capacity by self-learning and correcting itself through the records of past human evaluations.

Moreover, a modular architectural design was structured for the tool with 4 main modules. The pre-processing module works on the raw corpus and produces a pre-processed corpus for the extraction module. The extraction module has two parts, the syntactic and statistical extraction. The syntactic extraction works on the pre-processed corpus and produces a list of candidate MWEs. This list of candidate MWEs is then utilized by the statistical extraction to produce a list of filtered MWEs. The list of MWEs is presented to the evaluation module that produces a list of validated MWEs. It is the validated list that is lexically encoded by the encoding module and converted to XML-LMF lexicon.

In the design stage, HTML and CSS are used for the frontend, while PHP and Python are used for the backend with MySQL as the database. BNC was the corpus from which the frequency count for words were derived for the various AMs used for the statistical extraction.

For evaluation of the tool, various texts were processed and evaluated by 2 human evaluators. The precision rate usually ranged from 0.39 to 0.75, while the recall rate ranged from 0.64 to 0.96.

6.2 Conclusion

This research built a web-based tool that can automatically discover Nominal MWEs in large non-annotated English text, then present these extracted MWEs to human evaluators for validation and then lexically encode the validated MWEs in a lexicon that could be used by NLPs. To ensure easy integration of the lexicon into NLP applications, the XML-LMF format was used.

The evaluation of the tool reveals that the precision rate and recall rate of the statistical extraction method depends partially on the nature of the base corpus used for the frequency count of words for the association measures. That is, if a nominal MWE is totally absent from the base corpus, it is likely to be dropped by statistical extraction. Naturally, a base corpus that is current and expanding to include new MWEs is desirable for statistical extraction. Nevertheless, such corpus would become bigger in size and demand more computing power to process, unless the frequency count could be derived directly and automatically online from a more powerful computer. Such phenomenon still does not guarantee that statistical extraction would

become perfect because the various AMs only measure collocation strength and not all collocations are MWEs. Collocation, albeit important, is not the only feature of MWEs as there are other features of non-compositionality, ambiguity, among others. This necessitate the use of human evaluation before an MWE is added to the lexicon.

Human evaluation is sine qua non for automatic MWE extraction which usually record less than 100% accuracy. Automatic MWE discovery could not benefit NLPs directly or be plugged into NLPs directly. They have to go through human evaluation into a lexicon that could then be used by NLPs.

Therefore, proper discovery of automatic MWE to yield encoded lexicon could assist to speed up the work of lexicographers. In essence, this designed tool would help in this capacity, with its self-learning ability to reduce the work of lexicographers with each application.

6.3 Recommendations for Further Research

First and foremost, it would be helpful to statistical MWE extraction method if there is a project to provide frequency counts of words and terms from an online web service. It should also provide the total count of words in the ever-expanding base corpus at every given time. Certainly, such a project could not be undertaken simply by an individual, but probably a consortium.

This project only explored extracting Nominal MWEs from texts. Further research could be done to include other types of MWEs such as Verb-Particle Constructions, Light-Verb Constructions, Idioms (both decomposable and non-decomposable idioms) as well as fixed MWEs.

In this research, the recommendations in literature was followed for the contents and format of the MWE lexicon. Further research could be done to ascertain the real impact of such output lexicon on NLP applications. The findings could then be used to tweak its contents or format for better impact.

Chapter Seven: Reflection

Managing the references is always a daunting experience for me and probably for many students. The IEEE Referencing Style is particularly challenging because of the numbering system that is easy to misnumber.

This is the first time I am using the IEEE referencing for a document of this size. This made me research how to handle this more efficiency and accurately. I discovered that right inside Microsoft Word, there is the referencing facility. Under the References ribbon, there is the Citations and Bibliography group. With this facility, citations could be added under different categories: Book, Book Section, Journal Article, Article in a Periodical, Conference Proceedings, Report, Web site, among others. The facility then takes care of the numbering system. At the end, I could easily generate the Reference list. The beautiful thing is that the referencing style can be changed at any time with just one button and the whole referencing is automatically regenerated. What is more? The citations added in a particular document are available for use in another document in the same computer, so there is no need to add citations repeatedly. The central list of citations is available for editing and deleting. However, one needs to remember to update the reference list before printing out the document.

This inbuilt referencing facility might seem small but it is actually a lot for me and helped in a tremendous way. I called up 3 of my friends writing their M.Sc. theses at the time to ask if they knew of this facility in Microsoft Word and they did not. I taught them and they are still very grateful. It is amazing how we could be using a software for years or a hardware for years and not know many of the wonderful capabilities in them. I am resolved now to make more effort to learn more about things I use often. Certainly, this efficient way of managing referencing would help me in future research writing.

References

- [1] B. Dickson, "What is Natural Language Processing (NLP)?," *PC Magazine*, pp. 122-127, Feb 2020.
- [2] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, Cambridge, MA: The MIT Press, 1999.
- [3] J. Eisenstein, *Introduction to Natural Language Processing*, Cambridge M.A.: The MIT Press, 2019.
- [4] D. Jurafsky and J. H. Martin, *SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed. ed., New Jersey: Pearson Prentice Hall, 2008.
- [5] C. Ramisch, *Multiword Expressions Acquisition: A Generic and Open Framework*, New York: Springer, 2015.
- [6] M. Constant, G. Eryigit, J. Monti, L. Van der Plas, C. Ramisch, M. Rosner and A. Todirascu, "Multiword Expression Processing: A Survey," *Computational Linguistics*, vol. 43, no. 4, p. 837–892, 2017.
- [7] A. Gelbukh and O. Kolesnikova, "Multiword Expressions in NLP: General Survey and a Special Case of Verb-Noun Constructions," in *Emerging Applications of Natural Language Processing: Concepts and New Research*, S. Bandyopadhyay, S. K. Naskar and A. Ekbal, Eds., Hershey, IGI Global, 2013, pp. 1 - 21.
- [8] S. Koeva, "Multi-word Term Extraction for Bulgarian," in *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, Prague, Czech Republic, 2007.
- [9] B. Broda, M. Derwojedowa and M. Piasecki, "Recognition of Structured Collocations in an Inflective Language," *Systems Science*, vol. 34, no. 4, pp. 27-36, 2008.
- [10] C. Hore, M. Asahara and Y. Matsumoto, "Automatic Extraction of Fixed Multiword Expressions," in *International Conference on Natural Language Processing – IJCNLP 2005*, Jeju Island, Korea, 2005.
- [11] S. Agrawal, R. Sanyal and S. Sanyal, "Hybrid Method for Automatic Extraction of Multiword Expressions," *International Journal of Engineering & Technology*, vol. 7, no. 2.6, pp. 33-38, March 2018.
- [12] S. S. L. Piao, P. Rayson, D. Archer, A. Wilson and T. McEnery, "Extracting Multiword Expressions with A Semantic Tagger," in *Workshop on Multiword Expressions: Analysis, Acquisition and Treatment, at ACL 2003, 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 2003.

- [13] M. Weller and U. Heid, "Extraction of German Multiword Expressions from Parsed Corpora Using Context Features," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010.
- [14] A. Savary, S. Cordeiro and C. Ramisch, "Without lexicons, multiword expression identification will never fly: A position statement," in *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, Florence, Italy, 2019.
- [15] SIGLEX-MWE, "Joint Workshop on Multiword Expressions and Electronic Lexicons (MWE-LEX 2020)," 2020. [Online]. Available: http://multiword.sourceforge.net/PHITE.php?sitesig=CONF&page=CONF_02_MWE-LEX_2020__lb__COLING__rb__. [Accessed 29 February 2020].
- [16] D. Biber, S. Johansson, G. Leech, S. Conrad and E. Finegan, *Longman Grammar of Spoken and Written English*, 1st Edition ed., Harlow: Pearson Education, 1999.
- [17] P. Gantar, L. Colman, C. P. Escartin and H. M. Alonso, "Multiword Expressions: Between Lexicography and NLP," *International Journal of Lexicography*, vol. 32, no. 2, pp. 138 - 162, 2019.
- [18] N. Calzolari, C. J. Fillmore, R. Grishman, N. Ide, A. Lenci, C. MacLeod and A. Zampolli, "Towards Best Practice for Multiword Expressions in Computational Lexicons," Las Palmas, Canary Islands, 2002.
- [19] B. Krenn, "The Usual Suspects : Data-Oriented Models for Identification and Representation of Lexical Collocations," *Saarbrücken Dissertations in Computational Linguistics and Language Technology*, vol. 7, 2000.
- [20] S. Bird, E. Klein and E. Loper, *Natural Language Processing with Python*, Sebastopol, CA: O'Reilly Media Inc., 2009.
- [21] T. Briscoe, "Introduction to Linguistics for Natural Language Processing," 8 October 2013. [Online]. Available: <https://www.cl.cam.ac.uk/teaching/1314/L100/introoling.pdf>. [Accessed 8 September 2020].
- [22] T. Baldwin and S. N. Kim, "Multiword Expressions," in *Handbook of Natural Language Processing*, Second Edition ed., N. Indurkha and F. J. Damerau, Eds., London, CRC Press, 2010, p. 267–292.
- [23] M. Riedl and C. Biemann, "Impact of MWE resources on multiword recognition," in *Proceedings of the 12th Workshop on Multiword Expressions*, Berlin, Germany, 2016.
- [24] EAGLES: Expert Advisory Group on Language Engineering Standards, "ISLE: International Standards for Language Engineering," 2 March 2004. [Online]. Available: <http://www.ilc.cnr.it/EAGLES96/isle/>. [Accessed 25 July 2020].

- [25] Department of Computer Science, Vassar College, "XMELLT: Cross-lingual Multi-word Expression Lexicons for Language Technology," 2000. [Online]. Available: <https://www.cs.vassar.edu/~ide/XMELLT.html>. [Accessed 25 July 2020].
- [26] A. Savary, C. Ramisch, S. Cordeiro, F. Sangati, V. Vincze, B. QasemiZadeh, M. Candito, F. Cap, V. Giouli, I. Stoyanova and A. Doucet, "The PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions," in *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, Valencia, Spain, 2017.
- [27] A. Villavicencio, A. Copestake, B. Waldron and F. Lambeau, "Lexical Encoding of MWEs," in *Proceedings of the Workshop on Multiword Expressions: Integrating Processing*, Barcelona, Spain, 2004.
- [28] A. Copestake, F. Lambeau, A. Villavicencio, F. Bond, T. Baldwin, I. A. Sag and D. Flickinger, "Multiword expressions: linguistic precision and reusability," in *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain, 2002.
- [29] E. Bejček, P. Straňák and D. Zeman, "Influence of Treebank Design on Representation of Multiword Expressions," in *12th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2011)*, Tokyo, Japan, 2011.
- [30] J. Odijk, "Identification and Lexical Representation of Multiword Expressions," in *Essential Speech and Language Technology for Dutch: Results by the STEVIN programme*, P. Spyns and J. Odijk, Eds., Berlin, Heidelberg, Springer Berlin Heidelberg, 2013, pp. 201-217.
- [31] T. Lichte, S. Petitjean, A. Savary and J. Waszczuk, "Lexical encoding formats for multi-word expressions: The challenge of "irregular" regularities," in *Representation and parsing of multiword expressions: Current trends*, Berlin, Language Science Press, 2019, p. 1-33.
- [32] I. A. Sag, T. Baldwin, F. Bond, A. Copestake and D. Flickinger, "Multiword expressions: A pain in the neck for NLP," in *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, Mexico City, Mexico, 2002.
- [33] Universal Dependencies contributors, "Universal Dependencies," 2014. [Online]. Available: <https://universaldependencies.org/>. [Accessed 15 August 2020].
- [34] G. Francopoulo, M. George, N. Calzolari, M. Monachini, N. Bel, M. Pet and C. Soria, "Lexical Markup Framework (LMF)," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, 2006.
- [35] G. Francopoulo, N. Bel, M. George, N. Calzolari, M. Monachini, M. Pet and C. Soria, "Lexical Markup Framework (LMF) for NLP Multilingual Resources," *Language Resources and Evaluation*, vol. 43, no. 1, pp. 57-70, 2009.

- [36] M. Yazdani, M. Farahmand and J. Henderson, "Learning Semantic Composition to Detect Non-compositionality of Multiword Expressions," in *2015 Conference on Empirical Methods in Natural Language Processing: Association for Computational Linguistics*, Lisbon, 2015.
- [37] M. Svensson, "A very complex criterion of fixedness: Non-compositionality," in *Phraseology: An interdisciplinary perspective*, S. Granger and F. Meunier, Eds., Amsterdam, John Benjamins Publishing Company, 2008, p. 81–93.
- [38] M. T. Pazienza, M. Pennacchiotti and F. M. Zanzotto, "Terminology Extraction: An Analysis of Linguistic and Statistical Approaches," in *Knowledge Mining: Proceedings of the Nemis 2004 Final Conference (Studies in Fuzziness and Soft Computing)*, Berlin, 2004.
- [39] S. Reddy, D. McCarthy and S. Manandhar, "An Empirical Study on Compositionality in Compound Nouns," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand, 2011.
- [40] "Word Association Norms, Mutual Information, and Lexicography," *Computational Linguistics*, vol. 16, no. 4, p. 22–29, 1990.
- [41] P. Pecina, "A Machine Learning Approach to Multiword Expression Extraction," in *Proceedings of the LREC 2008 Workshop: Towards a Shared Task for Multiword Expressions*, Marrakech, Morocco, 2008.
- [42] O. Kolesnikova, "Survey of Word Co-occurrence Measures for Collocation Detection," *Computacion y Sistemas*, vol. 20, no. 3, pp. 327-344, September 2016.
- [43] F. Keller and M. Lapata, "Using the Web to Obtain Frequencies for Unseen Bigrams," *Computational Linguistics*, vol. 29, no. 3, p. 459–484, 2003.
- [44] S. Vintar, "Bilingual term recognition revisited: The bag-of-equivalents term alignment approach and its evaluation," *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, vol. 16, no. 2, pp. 141 - 158, 2010.
- [45] R. Stanković, C. Krstev, I. Obradović, B. Lazić and A. Trtovac, "Rule-based Automatic Multi-Word Term Extraction and Lemmatization," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia, 2016.
- [46] SPACY, "spaCy: Industrial-Strength Natural Language Processing In Python," 2016. [Online]. Available: <https://spacy.io/>. [Accessed 25 July 2020].
- [47] NLTK, "NLTK 3.5 documentation: Natural Language Toolkit," 13 April 2020. [Online]. Available: <http://www.nltk.org/>. [Accessed 25 July 2020].
- [48] S. Bird, E. Loper and E. Klein, *Natural Language Processing with Python*, Sebastopol, California: O'Reilly Media Inc., 2009.

- [49] J. D. Choi, J. Tetreault and A. Stent, "It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China, 2015.
- [50] StanfordNLP, "StanfordNLP 0.2.0 - Python NLP Library for Many Human Languages," 2018. [Online]. Available: <https://stanfordnlp.github.io/stanfordnlp/>. [Accessed 15 August 2020].
- [51] Stanford NLP Group, "Using CoreNLP within other programming languages and packages," 2020. [Online]. Available: <https://stanfordnlp.github.io/CoreNLP/other-languages.html#python>. [Accessed 15 August 2020].
- [52] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, Maryland, 2014.
- [53] P. Qi, T. Dozat, Y. Zhang and C. D. Manning, "Universal Dependency Parsing from Scratch," in *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium, 2018.
- [54] P. Qi, Y. Zhang, Y. Zhang, J. Bolton and C. D. Manning, "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Online, 2020.
- [55] Stanford NLP Group, "Tokenization & Sentence Segmentation: Use spaCy for Fast Tokenization and Sentence Segmentation," 2020. [Online]. Available: <https://stanfordnlp.github.io/stanza/tokenize.html#use-spacy-for-fast-tokenization-and-sentence-segmentation>. [Accessed 15 August 2020].
- [56] A. Villavicencio, V. Kordoni, Y. Zhang, M. Idiart and C. Ramisch, "Validation and Evaluation of Automatically Acquired Multiword Expressions for Grammar Engineering," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, 2007.
- [57] M. R. Costa-jussà, V. Daudaravicius and R. E. Banchs, "Integration of statistical collocation segmentations in a phrase-based statistical machine translation system," in *Proceedings of EAMT 2010 - 14th Annual Conference of the European Association for Machine Translation*, Saint-Raphaël, 2010.
- [58] M. Saunders, P. Lewis and A. Thornhill, *Research Methods for Business Students*, 7th ed., London: Pearson, 2016.
- [59] M. Crotty, *The Foundations of Social Research: Meaning and Perspective in the Research Process*, Newbury Park, California: SAGE Publications Ltd, 1998.

- [60] L. Cohen, L. Manion and K. Morrison, Research methods in education, 7th edition ed., London: Routledge, 2011.
- [61] R. Snieder and K. Lerner, The Art of Being a Scientist: A Guide for Graduate Students and their Mentors., Cambridge: Cambridge University Press, 2009.
- [62] C. R. Kothari, Research methodology: methods and techniques., New Delhi: New Age International, 2004.
- [63] D. Silverman, Doing Qualitative Research: A practical handbook., London: Sage, 2013.
- [64] Oxford Text Archive, "British National Corpus," 2015. [Online]. Available: <http://www.natcorp.ox.ac.uk/>. [Accessed 15 September 2020].
- [65] Oxford University Press, "British National Corpus (BNC)," [Online]. Available: <https://www.english-corpora.org/bnc/>. [Accessed 25 March 2020].
- [66] K. Roebuck, Systems Development Life Cycle (Sdlc) : High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors, Samford: Tebbo, 2011.
- [67] R. K. Wysocki, Effective Software Project Management, Indianapolis: John Wiley & Sons, 2010.
- [68] V. Osetskyi, "SDLC Models Explained: Agile, Waterfall, V-Shaped, Iterative, Spiral," 29 August 2017. [Online]. Available: <https://medium.com/existek/sdlc-models-explained-agile-waterfall-v-shaped-iterative-spiral-e3f012f390c5>. [Accessed 3 September 2020].
- [69] Existek, "SDLC Models Explained: Agile, Waterfall, V-Shaped, Iterative, Spiral," 4 August 2020. [Online]. Available: <https://existek.com/blog/sdlc-models/>. [Accessed 4 September 2020].
- [70] International Organization for Standardization (ISO), "ISO 24613:2008 Language resource management - Lexical markup framework (LMF)," 2008. [Online]. Available: <https://www.iso.org/standard/37327.html>. [Accessed 22 September 2020].
- [71] G. Francopoulo and M. George, "Model Description," in *LMF Lexical Markup Framework*, G. Francopoulo and P. Paroubek, Eds., Hoboken, New Jersey, John Wiley & Sons, Inc., 2013, pp. 19-40.
- [72] BNC Consortium, "British National Corpus, XML edition, Oxford Text Archive," 2007. [Online]. Available: <http://hdl.handle.net/20.500.12024/2554>. [Accessed 15 September 2020].
- [73] Explosion AI, "Install spaCy," 2016. [Online]. Available: <https://spacy.io/usage>. [Accessed 15 October 2020].

[74] English-Corpora.org, "iWeb: The 14 Billion Word Web Corpus," 2017. [Online].
Available: <https://www.english-corpora.org/iweb/>. [Accessed 16 October 2020].

APPENDICES

Appendix 1: Database Dump – to create tables used by the tool

```
--
-- Table structure for table `dtext`
--

CREATE TABLE `dtext` (
  `sn` int(11) NOT NULL,
  `dtext` text
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `dtext`
--

INSERT INTO `dtext` (`sn`, `dtext`) VALUES
(1, '');

-----

--
-- Table structure for table `lexicon`
--

CREATE TABLE `lexicon` (
  `sn` int(11) NOT NULL,
  `mwe` varchar(250) DEFAULT NULL,
  `archived` varchar(3) NOT NULL DEFAULT 'Yes',
  `mwedes` varchar(250) DEFAULT NULL,
  `compdes` varchar(250) DEFAULT NULL,
  `combpas` varchar(250) DEFAULT NULL,
  `mwemeaning` text,
  `mwesample` text
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-----

--
-- Table structure for table `mwes`
--

CREATE TABLE `mwes` (
  `sn` int(11) NOT NULL,
  `mwe` varchar(250) DEFAULT NULL,
  `pmi` decimal(10,2) DEFAULT NULL,
  `md` decimal(10,2) DEFAULT NULL,
  `dc` decimal(10,2) DEFAULT NULL,
  `archived` varchar(3) NOT NULL DEFAULT 'Yes',
  `choice1` varchar(3) NOT NULL DEFAULT 'No',
  `choice2` varchar(3) NOT NULL DEFAULT 'No',
  `mwedes` varchar(250) DEFAULT NULL,
  `compdes` varchar(250) DEFAULT NULL,
  `combpas` varchar(250) DEFAULT NULL,
  `mwemeaning` text,
  `mwesample` text
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```

-----
--
-- Table structure for table `mwes_temp`
--

CREATE TABLE `mwes_temp` (
  `sn` int(11) NOT NULL,
  `mwe` varchar(250) DEFAULT NULL,
  `pmi` decimal(10,2) DEFAULT NULL,
  `md` decimal(10,2) DEFAULT NULL,
  `dc` decimal(10,2) DEFAULT NULL,
  `archived` varchar(3) NOT NULL DEFAULT 'Yes',
  `choice1` varchar(3) NOT NULL DEFAULT 'No',
  `choice2` varchar(3) NOT NULL DEFAULT 'No',
  `mwedes` varchar(250) DEFAULT NULL,
  `compdes` varchar(250) DEFAULT NULL,
  `combat` varchar(250) DEFAULT NULL,
  `mwemeaning` text,
  `mwesample` text
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-----
--
-- Table structure for table `not_lexicon`
--

CREATE TABLE `not_lexicon` (
  `sn` int(11) NOT NULL,
  `mwe` varchar(250) DEFAULT NULL,
  `archived` varchar(3) NOT NULL DEFAULT 'Yes',
  `mwedes` varchar(250) DEFAULT NULL,
  `compdes` varchar(250) DEFAULT NULL,
  `combat` varchar(250) DEFAULT NULL,
  `mwemeaning` text,
  `mwesample` text
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Indexes for dumped tables
--

--
-- Indexes for table `dtext`
--
ALTER TABLE `dtext`
  ADD PRIMARY KEY (`sn`);

--
-- Indexes for table `lexicon`
--
ALTER TABLE `lexicon`
  ADD PRIMARY KEY (`sn`),
  ADD UNIQUE KEY `mwe` (`mwe`);

--
-- Indexes for table `mwes`
--
ALTER TABLE `mwes`

```

```

ADD PRIMARY KEY (`sn`);

--
-- Indexes for table `mwes_temp`
--
ALTER TABLE `mwes_temp`
  ADD PRIMARY KEY (`sn`);

--
-- Indexes for table `not_lexicon`
--
ALTER TABLE `not_lexicon`
  ADD PRIMARY KEY (`sn`),
  ADD UNIQUE KEY `mwe` (`mwe`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `dtext`
--
ALTER TABLE `dtext`
  MODIFY `sn` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `lexicon`
--
ALTER TABLE `lexicon`
  MODIFY `sn` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `mwes`
--
ALTER TABLE `mwes`
  MODIFY `sn` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `mwes_temp`
--
ALTER TABLE `mwes_temp`
  MODIFY `sn` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `not_lexicon`
--
ALTER TABLE `not_lexicon`
  MODIFY `sn` int(11) NOT NULL AUTO_INCREMENT;

```

Appendix 2: pos100.py - To run only pre-processing on Text

```

import spacy
import sys

nlp = spacy.load("en_core_web_sm")

try:
    dnumber = len(sys.argv[1:])
    data = sys.argv[1:]
    dtext = str(sys.argv[1:])
except:
    print (" Error")
    sys.exit(1)

```

```

#data is an array, a list. This line converts it to a string with space as the delimiter
text = ' '.join(map(str, data))

doc = nlp(text)
#remove stop words and punctuations
doc_cleaned = [token for token in doc if not token.is_stop and not token.is_punct]

for token in doc_cleaned:
    print (str(token) + ' [{0}]' + str(token.pos_) + ' [{0}]' + str(token.tag_) + ' [{0}]' +
          str(token.lemma_) + ' [{0}]' + str(spacy.explain(token.tag_)) + ' [{0}]' + str(token.dep_) + ' [{0}]'
          + str(token.head.text) + ' [{0}]' + str(spacy.explain(token.dep_)))

```

Appendix 3: pos200.py - To run pre-processing and syntactic extraction on Text

```

import spacy
import sys
from spacy.matcher import Matcher

nlp = spacy.load("en_core_web_sm")

try:
    dnumber = len(sys.argv[1:])
    data = sys.argv[1:]
    dtext = str(sys.argv[1:])
except:
    print ("Error")
    sys.exit(1)

text = ' '.join(map(str, data))

matcher = Matcher(nlp.vocab)

def extract_nominal_mwe(nlp_doc):
    proper = []
    pattern1 = [{"DEP": "compound", "POS": "NOUN"}, {"TAG": "HYPH", "OP": "?"}, {"POS": "NOUN"}]
    pattern2 = [{"DEP": {"IN": ["compound", "amod"]}, "TAG": "JJ"}, {"TAG": "HYPH", "OP": "?"}, {"DEP": "compound", "POS": "NOUN", "OP": "*"}, {"POS": "NOUN"}]
    pattern3 = [{"DEP": {"IN": ["conj", "acl"]}, "TAG": "VBG"}, {"TAG": "HYPH", "OP": "?"}, {"POS": "NOUN"}]
    pattern4 = [{"DEP": {"IN": ["compound", "amod"]}, "TAG": "VB"}, {"TAG": "HYPH", "OP": "?"}, {"POS": "NOUN"}]
    pattern5 = [{"POS": "NOUN"}, {"TAG": "HYPH"}, {"TAG": "IN"}, {"TAG": "HYPH"}, {"POS": "NOUN"}]
    pattern6 = [{"DEP": {"IN": ["compound", "conj"]}, "POS": "NOUN"}, {"TAG": "HYPH", "OP": "?"}, {"TAG": "JJ"}]
    pattern = [{"POS": "PROPN", "OP": "+"}, {"LOWER": {"IN": ["for", "of", "in"]}, "OP": "?"}, {"POS": "NOUN", "OP": "*"}, {"POS": "PROPN", "OP": "+"}]
    matcher.add("FULL_NAME", None, pattern)
    matcher.add("NOUN_NOUN", None, pattern1)
    matcher.add("ADJ_NOUN", None, pattern2)
    matcher.add("VERB_NOUN", None, pattern3)
    matcher.add("VERB_NOUN2", None, pattern4)
    matcher.add("NOUN_PREP", None, pattern5)
    matcher.add("NOUN_ADJ", None, pattern6)

    matches = matcher(nlp_doc)
    for match_id, start, end in matches:
        span = nlp_doc[start:end]
        proper.append(span.text)

```

```

        return proper

doc = nlp(text)
doc_cleaned = [token for token in doc if not token.is_stop and not token.is_punct]

result = extract_nominal_mwe(doc)

#Remove duplicates
result = list(dict.fromkeys(result))

#Reducing it to unique names
lily = result
feli = ' | '.join(result)
wumi = []
n = len(lily)
i = 0

while i < n:
    if feli.count(lily[i]) == 1:
        wumi.append(lily[i])
        i += 1

print(wumi)

```

Appendix 4: pos300.py - To run pre-processing, syntactic and statistical extraction on Text

```

import spacy
import sys
from spacy.matcher import Matcher
import pickle
from collections import Counter
import math
import mysql.connector

nlp = spacy.load("en_core_web_sm")
nlp2 = spacy.load("en_core_web_sm", disable=["parser", "ner"])

def d_pmi(x,y,xy,b):
    try:
        z = (xy/b) / ((x/b)*(y/b))
        a = math.log(z)
    except:
        a = 0
    a = round(a,3)
    return a

def d_md(x,y,xy,b):
    try:
        z = ((xy**2)/b) / ((x/b)*(y/b))
        a = math.log(z)
    except:
        a = 0
    a = round(a,3)
    return a

def d_dc(x,y,xy):
    try:
        a = (xy*2) / (x+y)

```

```

except:
    a = 0
a = round(a,3)
return a

#Load single word frequency file
try:
    the_filename = "BNC_Frequency.txt"
    with open(the_filename, 'rb') as fa:
        word_freq = pickle.load(fa)
except:
    print (" Error loading file1 ", flush=True)
    sys.exit(1)

fa.close()

#Load bigram frequency file
try:
    the_filename = "BNC_Frequency_BiGram.txt"
    with open(the_filename, 'rb') as fa:
        bi_freq = pickle.load(fa)
except:
    print (" Error loading file2 ", flush=True)
    sys.exit(1)

fa.close()

#Getting Text from PHP
try:
    dnnumber = len(sys.argv[1:])
    data = sys.argv[1:]
    dtext = str(sys.argv[1:])
except:
    print (" Error")
    sys.exit(1)

text = ' '.join(map(str, data))

tw = word_freq["DTotaWords2020"]

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="mwe"
)
mycursor = mydb.cursor()
mycursor.execute("DELETE FROM mwes")

matcher = Matcher(nlp.vocab)

def extract_nominal_mwe(nlp_doc):
    pattern1 = [{"DEP": 'compound', 'POS': 'NOUN'}, {'TAG': 'HYPH', 'OP': '?'}, {'POS': 'NOUN'}]
    pattern2 = [{"DEP": 'compound', 'TAG': 'JJ'}, {'TAG': 'HYPH', 'OP': '?'}, {'POS': 'NOUN'}]
    pattern3 = [{"DEP": 'amod', 'TAG': 'JJ'}, {'TAG': 'HYPH', 'OP': '?'}, {'POS': 'NOUN'}]
    pattern4 = [{"DEP": 'conj', 'TAG': 'VBG'}, {'TAG': 'HYPH', 'OP': '?'}, {'POS': 'NOUN'}]
    pattern5 = [{"DEP": 'acl', 'TAG': 'VBG'}, {'TAG': 'HYPH', 'OP': '?'}, {'POS': 'NOUN'}]
    pattern6 = [{"DEP": 'compound', 'TAG': 'VB'}, {'TAG': 'HYPH', 'OP': '?'}, {'POS': 'NOUN'}]
    pattern7 = [{"DEP": 'amod', 'TAG': 'VB'}, {'TAG': 'HYPH', 'OP': '?'}, {'POS': 'NOUN'}]
    pattern8 = [{"DEP": 'compound', 'POS': 'NOUN'}, {'TAG': 'HYPH', 'OP': '?'}, {'TAG': 'JJ'}]
    pattern9 = [{"DEP": 'conj', 'POS': 'NOUN'}, {'TAG': 'HYPH', 'OP': '?'}, {'TAG': 'JJ'}]

```

```

pattern10 = [{'POS': 'NOUN'}, {'TAG': 'HYPH'}, {'TAG': 'IN'}, {'TAG': 'HYPH'}, {'POS': 'NOUN'}]
pattern11 = [{'DEP': 'compound', 'TAG': 'JJ'}, {'TAG': 'HYPH', 'OP': '?'}, {'DEP': 'compound', 'POS':
'NOUN', 'OP': '+'}, {'POS': 'NOUN'}]
pattern12 = [{'DEP': 'amod', 'TAG': 'JJ'}, {'TAG': 'HYPH', 'OP': '?'}, {'DEP': 'compound', 'POS':
'NOUN', 'OP': '+'}, {'POS': 'NOUN'}]
pattern = [{'POS': 'PROPN', 'OP': '+'}, {'LOWER': 'IN': ['for', 'of', 'in']}, {'OP': '?'}, {'POS': 'NOUN',
'OP': '*'}, {'POS': 'PROPN', 'OP': '+'}]
matcher.add('FULL_NAME', None, pattern)
matcher.add('NOUN_NOUN', None, pattern1)
matcher.add('ADJ_NOUN1', None, pattern2)
matcher.add('ADJ_NOUN2', None, pattern3)
matcher.add('VERB_NOUN1', None, pattern4)
matcher.add('VERB_NOUN2', None, pattern5)
matcher.add('VERB_NOUN3', None, pattern6)
matcher.add('VERB_NOUN4', None, pattern7)
matcher.add('NOUN_ADJ1', None, pattern8)
matcher.add('NOUN_ADJ2', None, pattern9)
matcher.add('NOUN_PREP', None, pattern10)
matcher.add('ADJ_NOUN3', None, pattern11)
matcher.add('ADJ_NOUN4', None, pattern12)

matches = matcher(nlp_doc)
for match_id, start, end in matches:
    string_id = nlp.vocab.strings[match_id]
    span = nlp_doc[start:end]

    if string_id=="NOUN_NOUN":
        compat = "NOUN(compound)->NOUN"
    elif string_id=="ADJ_NOUN1":
        compat = "JJ(compound)->NOUN"
    elif string_id=="ADJ_NOUN2":
        compat = "JJ(amod)->NOUN"
    elif string_id=="VERB_NOUN1":
        compat = "VBG(conj)->NOUN"
    elif string_id=="VERB_NOUN2":
        compat = "VBG(aci)->NOUN"
    elif string_id=="VERB_NOUN3":
        compat = "VB(compound)->NOUN"
    elif string_id=="VERB_NOUN4":
        compat = "VB(amod)->NOUN"
    elif string_id=="NOUN_ADJ1":
        compat = "NOUN(compound)->JJ"
    elif string_id=="NOUN_ADJ2":
        compat = "NOUN(conj)->JJ"
    elif string_id=="NOUN_PREP":
        compat = "NOUN-in-NOUN"
    elif string_id=="FULL_NAME":
        compat = "PROPER_NAME"
    elif string_id=="ADJ_NOUN3":
        compat = "JJ(compound)->NOUN(compound)->NOUN"
    elif string_id=="ADJ_NOUN4":
        compat = "JJ(amod)->NOUN(compound)->NOUN"

mwe = span.text
mwepos = "Noun Compound"
mwecon = "True"

mm = mwe.split()
if len(mm)>1:
    if len(mm)==2:
        w1 = word_freq[mm[0]]

```

```

w2 = word_freq[mm[1]]
w12 = bi_freq[mwe]
pmi = d_pmi(w1,w2,w12,tw)
md = d_md(w1,w2,w12,tw)
dc = d_dc(w1,w2,w12)

cle = []
cco = []
wdoc = nlp2(mwe)
for wtoken in wdoc:
    cle.append(wtoken.lemma_)
    cco.append(str(wtoken.lemma_) + ":" + str(wtoken.tag_))
mwelem = ''.join(cle)
compdes = ','.join(cco)
mwedes = "POS:" + str(mwepos) + ",LEMMA:" + str(mwelem) + ",CONTIGUOUS:" + str(mwecon)

fl="INSERT INTO mwes (mwe,pmi,md,dc,mwedes,compdes,combpas) VALUES (' " + str(mwe) +
"', " + str(pmi) + ", " + str(md) + ", " + str(dc) + ", " + str(mwedes) + ", " + str(compdes) + ", " + str(combpas) + ") "
mycursor.execute(fl)
else:
    cle = []
    cco = []
    wdoc = nlp2(mwe)
    for wtoken in wdoc:
        cle.append(wtoken.lemma_)
        cco.append(str(wtoken.lemma_) + ":" + str(wtoken.tag_))
    mwelem = ''.join(cle)
    compdes = ','.join(cco)
    mwedes = "POS:" + str(mwepos) + ",LEMMA:" + str(mwelem) + ",CONTIGUOUS:" + str(mwecon)

    fl="INSERT INTO mwes (mwe,pmi,md,dc,mwedes,compdes,combpas) VALUES (' " + str(mwe) +
    "',NULL,NULL,NULL, " + str(mwedes) + ", " + str(compdes) + ", " + str(combpas) + ") "
    mycursor.execute(fl)
else:
    cle = []
    cco = []
    wdoc = nlp2(mwe)
    for wtoken in wdoc:
        cle.append(wtoken.lemma_)
        cco.append(str(wtoken.lemma_) + ":" + str(wtoken.tag_))
    mwelem = ''.join(cle)
    compdes = ','.join(cco)
    mwedes = "POS:" + str(mwepos) + ",LEMMA:" + str(mwelem) + ",CONTIGUOUS:" + str(mwecon)

    fl="INSERT INTO mwes (mwe,pmi,md,dc,mwedes,compdes,combpas) VALUES (' " + str(mwe) +
    "',NULL,NULL,NULL, " + str(mwedes) + ", " + str(compdes) + ", " + str(combpas) + ") "
    mycursor.execute(fl)

proper = "Done"
return proper

doc = nlp(text)
result = extract_nominal_mwe(doc)

mycursor.execute("UPDATE mwes SET archived='No', choice1='Yes', choice2='Yes' where md > 3.5")
mycursor.execute("UPDATE mwes SET archived='No', choice1='Yes', choice2='Yes' where pmi IS
NULL")

#Remove duplicates
mycursor.execute("DELETE t1 FROM mwes t1 JOIN mwes t2 ON t2.mwe = t1.mwe AND t2.sn < t1.sn")

```

```

#Reducing it to unique names
mycursor.execute("DELETE FROM mwes_temp")
mycursor.execute("INSERT INTO mwes_temp SELECT * FROM mwes")

mycursor.execute("SELECT mwe FROM mwes")
myresult = mycursor.fetchall()
for x in myresult:
    y = ''.join(x)
    mys = "DELETE FROM mwes WHERE mwe = '" + str(y) + "' AND (SELECT count(*) FROM mwes_temp t1
WHERE t1.mwe LIKE '%" + str(y) + "%') > 1"
    mycursor.execute(mys)

mycursor.execute("DELETE FROM mwes_temp")

print(result)

```

Appendix 5: pos400.py - To generate the current XML-LMF file

```

from nltk.corpus import wordnet
import sys
import mysql.connector
import xml.etree.ElementTree as ET
import shutil

#Getting Text from PHP
try:
    dnumber = len(sys.argv[1:])
    data = sys.argv[1:]
    dtext = str(sys.argv[1:])
except:
    print (" Error")
    sys.exit(1)
text = ' '.join(map(str, data))

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="learn"
)
mycursor = mydb.cursor()
mycursor.execute("SELECT mwe,mwedes,compdes,combpas FROM mwes where choice1='Yes' and
choice2='Yes' order by mwe asc")
myresult = mycursor.fetchall()

myxml = ET.Element('LexicalResource')
myxml.set('dtdVersion', '15')
level1a = ET.SubElement(myxml, 'GlobalInformation')
level1a1a = ET.SubElement(level1a, 'feat')
level1a1a.set('att', 'languageCoding')
level1a1a.set('val', 'ISO 639-3')
level1b = ET.SubElement(myxml, 'Lexicon')
level1b1a = ET.SubElement(level1b, 'feat')
level1b1a.set('att', 'language')
level1b1a.set('val', 'eng')
for x in myresult:
    level1b1b = ET.SubElement(level1b, 'LexicalEntry')
    level1b1b1f = ET.SubElement(level1b1b, 'feat')
    level1b1b1f.set('att', 'Entry_Type')

```

```

level1b1b1f.set('val', 'Multiword')
jjj=x[0]
level1b1b1g = ET.SubElement(level1b1b, 'feat')
level1b1b1g.set('att', 'Written_Form')
level1b1b1g.set('val', jjj)
li=x[1].split(",")
level1b1b1a = ET.SubElement(level1b1b, 'MWE_Description')
fe=li[0].split(":")
feat1 = ET.SubElement(level1b1b1a, 'feat')
feat1.set('att', 'POS')
feat1.set('val', fe[1])
fe=li[1].split(":")
feat2 = ET.SubElement(level1b1b1a, 'feat')
feat2.set('att', 'Lemma')
feat2.set('val', fe[1])
fe=li[2].split(":")
feat3 = ET.SubElement(level1b1b1a, 'feat')
feat3.set('att', 'Contiguous')
feat3.set('val', fe[1])
level1b1b1b = ET.SubElement(level1b1b, 'Component_Description')
wu = x[2].split(",")
for y in wu:
    z = y.split(":")
    level1b1b1bC1 = ET.SubElement(level1b1b1b, 'Component')
    feat4 = ET.SubElement(level1b1b1bC1, 'feat')
    feat4.set('att', 'Lemma')
    feat4.set('val', z[0])
    feat5 = ET.SubElement(level1b1b1bC1, 'feat')
    feat5.set('att', 'POS')
    feat5.set('val', z[1])
level1b1b1c = ET.SubElement(level1b1b, 'Combination_Pattern')
feat6 = ET.SubElement(level1b1b1c, 'feat')
feat6.set('att', 'UD_Notation')
feat6.set('val', x[3])
level1b1b1d = ET.SubElement(level1b1b, 'MWE_Meaning')
try:
    syns1 = wordnet.synsets(x[0])
except:
    feat8 = ET.SubElement(level1b1b1d, 'feat')
    feat8.set('att', 'Meaning1')
    feat8.set('val', '')
else:
    nnn=len(syns1)
    if nnn > 0:
        s1 = 0
        for j in syns1:
            defin1 = j.definition()
            s1 = s1 + 1
            t1 = "Meaning" + str(s1)
            feat7 = ET.SubElement(level1b1b1d, 'feat')
            feat7.set('att', t1)
            feat7.set('val', defin1)
        else:
            feat8 = ET.SubElement(level1b1b1d, 'feat')
            feat8.set('att', 'Meaning1')
            feat8.set('val', '')
level1b1b1e = ET.SubElement(level1b1b, 'Sample_Sentences')
try:
    syns2 = wordnet.synsets(x[0])
except:
    feat10 = ET.SubElement(level1b1b1e, 'feat')

```

```

        feat10.set('att', 'Sample1')
        feat10.set('val', '')
    else:
        mmm=len(syns2)
        if mmm > 0:
            s2 = 0
            for q in syns2:
                defin2 = q.examples()
                defin3 = " || ".join(defin2)
                s2 = s2 + 1
                t2 = "Sample" + str(s2)
                feat9 = ET.SubElement(level1b1b1e, 'feat')
                feat9.set('att', t2)
                feat9.set('val', defin3)
            else:
                feat10 = ET.SubElement(level1b1b1e, 'feat')
                feat10.set('att', 'Sample1')
                feat10.set('val', '')
    b_xml = ET.tostring(myxml)

    try:
        the_filename = " /home/nisltg5/public_html/mwe/000/XML_LMF.xml "
        with open(the_filename, 'wb') as f:
            f.write(b_xml)
    except:
        print (" Error writing to file", flush=True)
        sys.exit(1)

    f.close()

    newPath = shutil.copy('/home/nisltg5/public_html/mwe/000/XML_LMF.xml',
        '/home/nisltg5/public_html/mwe/')
    print(" Done")

```

Appendix 6: pos500.py - To generate the cumulative XML-LMF file

```

from nltk.corpus import wordnet
import sys
import mysql.connector
import xml.etree.ElementTree as ET
import shutil

#Getting Text from PHP
try:
    dnumber = len(sys.argv[1:])
    data = sys.argv[1:]
    dtext = str(sys.argv[1:])
except:
    print (" Error")
    sys.exit(1)
text = ' '.join(map(str, data))

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="learn"
)
mycursor = mydb.cursor()
mycursor.execute(" SELECT mwe,mwedex,compdes,combpas FROM lexicon order by mwe asc")

```

```

myresult = mycursor.fetchall()

myxml = ET.Element('LexicalResource')
myxml.set('dtdVersion', '15')
level1a = ET.SubElement(myxml, 'GlobalInformation')
level1a1a = ET.SubElement(level1a, 'feat')
level1a1a.set('att', 'languageCoding')
level1a1a.set('val', 'ISO 639-3')
level1b = ET.SubElement(myxml, 'Lexicon')
level1b1a = ET.SubElement(level1b, 'feat')
level1b1a.set('att', 'language')
level1b1a.set('val', 'eng')

for x in myresult:
    level1b1b = ET.SubElement(level1b, 'LexicalEntry')
    level1b1b1f = ET.SubElement(level1b1b, 'feat')
    level1b1b1f.set('att', 'Entry_Type')
    level1b1b1f.set('val', 'Multiword')
    jjj=x[0]
    level1b1b1g = ET.SubElement(level1b1b, 'feat')
    level1b1b1g.set('att', 'Written_Form')
    level1b1b1g.set('val', jjj)
    li=x[1].split(",")
    level1b1b1a = ET.SubElement(level1b1b, 'MWE_Description')
    fe=li[0].split(":")
    feat1 = ET.SubElement(level1b1b1a, 'feat')
    feat1.set('att', 'POS')
    feat1.set('val', fe[1])
    fe=li[1].split(":")
    feat2 = ET.SubElement(level1b1b1a, 'feat')
    feat2.set('att', 'Lemma')
    feat2.set('val', fe[1])
    fe=li[2].split(":")
    feat3 = ET.SubElement(level1b1b1a, 'feat')
    feat3.set('att', 'Contiguous')
    feat3.set('val', fe[1])
    level1b1b1b = ET.SubElement(level1b1b, 'Component_Description')
    wu = x[2].split(",")
    for y in wu:
        z = y.split(":")
        level1b1b1bC1 = ET.SubElement(level1b1b1b, 'Component')
        feat4 = ET.SubElement(level1b1b1bC1, 'feat')
        feat4.set('att', 'Lemma')
        feat4.set('val', z[0])
        feat5 = ET.SubElement(level1b1b1bC1, 'feat')
        feat5.set('att', 'POS')
        feat5.set('val', z[1])
    level1b1b1c = ET.SubElement(level1b1b, 'Combination_Pattern')
    feat6 = ET.SubElement(level1b1b1c, 'feat')
    feat6.set('att', 'UD_Notation')
    feat6.set('val', x[3])
    level1b1b1d = ET.SubElement(level1b1b, 'MWE_Meaning')
    try:
        syns1 = wordnet.synsets(x[0])
    except:
        feat8 = ET.SubElement(level1b1b1d, 'feat')
        feat8.set('att', 'Meaning1')
        feat8.set('val', '')
    else:
        nnn=len(syns1)
        if nnn > 0:

```

```

s1 = 0
for j in syns1:
    defin1 = j.definition()
    s1 = s1 + 1
    t1 = "Meaning" + str(s1)
    feat7 = ET.SubElement(level1b1b1d, 'feat')
    feat7.set('att', t1)
    feat7.set('val', defin1)
else:
    feat8 = ET.SubElement(level1b1b1d, 'feat')
    feat8.set('att', 'Meaning1')
    feat8.set('val', '')
level1b1b1e = ET.SubElement(level1b1b, 'Sample_Sentences')
try:
    syns2 = wordnet.synsets(x[0])
except:
    feat10 = ET.SubElement(level1b1b1e, 'feat')
    feat10.set('att', 'Sample1')
    feat10.set('val', '')
else:
    mmm=len(syns2)
    if mmm > 0:
        s2 = 0
        for q in syns2:
            defin2 = q.examples()
            defin3 = " || ".join(defin2)
            s2 = s2 + 1
            t2 = "Sample" + str(s2)
            feat9 = ET.SubElement(level1b1b1e, 'feat')
            feat9.set('att', t2)
            feat9.set('val', defin3)
        else:
            feat10 = ET.SubElement(level1b1b1e, 'feat')
            feat10.set('att', 'Sample1')
            feat10.set('val', '')

b_xml = ET.tostring(myxml)
try:
    the_filename = "/home/nisltg5/public_html/mwe/000/XML_LMF_Cumulative.xml"
    with open(the_filename, 'wb') as f:
        f.write(b_xml)
except:
    print (" Error writing to file", flush=True)
    sys.exit(1)
f.close()

newPath = shutil.copy('/home/nisltg5/public_html/mwe/000/XML_LMF_Cumulative.xml',
'/home/nisltg5/public_html/mwe/')
print(" Done")

```

Appendix 7: pos600.py - To add missing MWE specified by human evaluators

```

import spacy
import sys
import mysql.connector

nlp2 = spacy.load("en_core_web_sm", disable=["parser", "ner"])

#Getting Text from PHP
try:

```

```

dnnumber = len(sys.argv[1:])
data = sys.argv[2:]
whos = sys.argv[1]
dtext = str(sys.argv[1:])
except:
    print (" Error")
    sys.exit(1)
text = ' '.join(map(str, data))
dwho = str(whos)

whichannotator = "choice"+dwho;
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="learn"
)
mycursor = mydb.cursor()
mycursor.execute("SELECT dtext FROM dtext where sn=1")
myresult = mycursor.fetchall()
otext = ''.join(myresult[0])

def process_nominal_mwe(nlp_doc):
    mwe = nlp_doc
    mwepos = "Noun Compound"
    mwecon = "True"
    combat = ""
    mm = mwe.split()
    cle = []
    cco = []
    wdoc = nlp2(mwe)
    for wtoken in wdoc:
        cle.append(wtoken.lemma_)
        cco.append(str(wtoken.lemma_) + ":" + str(wtoken.tag_))
    mwelem = ''.join(cle)
    compdes = ''.join(cco)
    mwedes = "POS:" + str(mwepos) + ",LEMMA:" + str(mwelem) + ",CONTIGUOUS:" + str(mwecon)
    fl="INSERT INTO mwes (mwe,pmi,md,dc,mwedes,compdes,combat,archived," + str(whichannotator)
    + ") VALUES ('" + str(mwe) + " ','NULL,NULL,NULL,'" + str(mwedes) + " ','" + str(compdes) + " ','" +
    str(combat) + " ','Yes', 'Yes') "
    mycursor.execute(fl)
    proper = "Done"
    return proper

dtext = text.split(",")
for eachitem in dtext:
    eacho = eachitem.strip()
    eachd = eacho.strip(",:.")
    nnn = len(eachd.split())
    if nnn > 1:
        if eachd in otext:
            mycursor.execute("SELECT * FROM mwes where mwe='"+str(eachd)+"'")
            myresult = mycursor.fetchall()
            ppp = len(myresult)
            if ppp < 1:
                result = process_nominal_mwe(eachd)

#Remove duplicates
mycursor.execute("DELETE t1 FROM mwes t1 JOIN mwes t2 ON t2.mwe = t1.mwe AND t2.sn < t1.sn")
print(result)

```

Appendix 8: index.php

```
<!doctype html>
<html>
<head>
<?php include(" head.php"); ?>
</head>
<body>
<?php include(" top.php"); ?>


<div style="text-align:center;margin-bottom:50px;padding-bottom:20px;border-bottom:1px solid #222;">
<h3>Add Text Below...</h3>
<form id="form1" name="form1" method="post" action="process.php" accept-charset="utf-8">
<textarea name="contenter" id="editor1" cols="95" rows="10"><?php
if(isset($_POST['contenter']) && $_POST['contenter']!=""){ echo($_POST['contenter']); }
?></textarea>


<div style="margin:10px 25px;">
    <label>
        <input type="radio" name="stage" value="pre" id="stage_0" checked>
        Preprocessing</label>
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
    <label>
        <input type="radio" name="stage" value="syn" id="stage_1">
        Syntactic Extraction</label>
        ~
    <label>
        <input type="radio" name="stage" value="sta" id="stage_2">
        Statistical Extraction</label>
</div>


<div>
<a href="index.php">Reset</a>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
<input name="addtext" type="submit" id="addtext" value=" Process Text " />
</div>
</form>
</div>


<div style="text-align:center;">
<a href="annotate1.php">Annotator 1</a>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
<a href="annotate2.php">Annotator 2</a>

&nbsp;&nbsp;&~
<a href="seeresult.php">See Result</a>
</div>


<?php include(" bottom.php"); ?>
<?php include(" foot.php"); ?>
</body>
</html>
```

Appendix 9: process.php

```
<!doctype html>
<html>
<head>
<?php include(" head.php"); ?>
```



```

<?php
$pyscript = escapeshellcmd(' /home/nisltg5/public_html/mwe/000/pos100.py ');
$python = escapeshellcmd('/opt/imh-python/bin/python3.7');
$pydata = escapeshellarg($dinput);

exec("python $pyscript $pydata", $result, $return);
?>
<p>&nbsp;</p>
<table class="table table-bordered table-hover table-striped table-condensed" style="width:75%;
margin:0px auto;">
<tr>
<th>S/N</th>
<th>TOKEN</th><th>POS</th><th>TAG</th><th>LEMMA</th><th>DESCRIPTION</th><th>DEPEN
DENCY</th><th>HEAD WORD</th><th>DESCRIPTION</th>
</tr>
<?php
$gv=0;

foreach($result as $row){
    $gv++;
    $line=explode("{{0}}", $row);
    echo "<tr><td>$gv</td>";
        foreach($line as $entry){
            echo "<td>" . $entry . "</td>";
        }
    echo "</tr>";
}
?>
</table>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

```

Appendix 11: syn.php

```

<?php
$pyscript = escapeshellcmd(' /home/nisltg5/public_html/mwe/000/pos200.py ');
$python = escapeshellcmd('/opt/imh-python/bin/python3.7');
$pydata = escapeshellarg($dinput);

exec("python $pyscript $pydata", $result, $return);
$g = implode(" | ", $result);

if(preg_match_all('/\'.+?\.'/, $g, $matches)) {
?>

<p>&nbsp;</p>
<table class="table table-bordered table-hover table-striped table-condensed" style="width:75%;
margin:0px auto;">
<tr>
<th>S/N</th>
<th>CANDIDATE NOMINAL MWE</th>
</tr>
<?php
$gv=0;

foreach ($matches[0] as $match) {
    $h = str_replace(" ", "", $match);
    $gv++;
}

```

```

        echo "<tr><td>$gv</td>";
        echo "<td>";
        echo $h;
        echo "</td>";
        echo "</tr>";
    }

```

```

?>
</table>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<?php
}
?>

```

Appendix 12: sta.php

```

<?php
$pyscript = escapeshellcmd(' /home/nislgt5/public_html/mwe/000/pos300.py ');
$python = escapeshellcmd(' /opt/imh-python/bin/python3.7 ');
$pydata = escapeshellarg($dinput);

exec(" $python $pyscript $pydata ", $result, $return);
include(" dataconn.php");

//Add the corpora to a table
$dtext = getst($_POST['contenter']);
$sql="update dtext set dtext=? where sn = 1 ";
$stmt=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmt,$sql)) {
    mysqli_stmt_bind_param($stmt,"s",$dtext);
    if(!mysqli_stmt_execute($stmt)){
        $bb="<p>Error: Could not execute query. " . mysqli_error($csn) . "</p>";
    } else {
        $bb="<p>Text Successfully Added!</p>";
    }
    mysqli_stmt_close($stmt);
} else {
    $bb="<p>Error: Could not prepare query. " . mysqli_error($csn) . "</p>";
}
?>

<?php
//Add to filtered MWE if the MWE exists in cumulative lexicon, i.e. previously selected by both annotators
$query="UPDATE mwes t1 JOIN lexicon t2 ON t1.mwe = t2.mwe SET t1.archived = 'No', choice1='Yes',
choice2='Yes';";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
    }
}
?>

<?php
//Remove from filtered MWE if the MWE exists in cumulative NOT lexicon, i.e. previously rejected by both
annotators

```

```

$query="UPDATE mwes t1 JOIN not_lexicon t2 ON t1.mwe = t2.mwe SET t1.archived = 'Yes',
choice1='No', choice2='No';";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
    }
}
?>

<h2 style="text-align:center">Filtered MWEs</h2>
<?php
$query="select sn,mwe,pmi,md,dc from mwes where archived='No' order by md desc";
$stmtu=mysqli_stmt_init($csn);
$num=0;
if (mysqli_stmt_prepare($stmtu,$query)) {
    //mysqli_stmt_bind_param($stmtu);

    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        $num = mysqli_stmt_num_rows($stmtu);
        mysqli_stmt_bind_result($stmtu, $usn, $umwe, $upmi, $umd, $udc);
    }
}

if ($num>0) { $gv=0;
?>
<p>&nbsp;</p>
<table class="table table-bordered table-hover table-striped table-condensed" style="width:75%;
margin:0px auto;">
<tr>
<th>S/N</th>
<th>CANDIDATE NOMINAL MWE</th>
<th>PMI</th>
<th>MD</th>
<th>DC</th>
</tr>

<?php while (mysqli_stmt_fetch($stmtu)) { $gv++; ?>
<tr>
<td><?php echo($gv); ?></td>
<td><?php echo $umwe; ?></td>
<td><?php echo $upmi; ?></td>
<td><?php echo $umd; ?></td>
<td><?php echo $udc; ?></td>
</tr>
<?php } ?>

</table>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<?php
}
?>

<h2 style="text-align:center">To be Dropped MWEs</h2>
<?php
$query="select sn,mwe,pmi,md,dc from mwes where archived='Yes' order by md desc";
$stmtu=mysqli_stmt_init($csn);
$num=0;

```

```

if (mysqli_stmt_prepare($stmtu,$query)) {
    //mysqli_stmt_bind_param($stmtu);

    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        $unum = mysqli_stmt_num_rows($stmtu);
        mysqli_stmt_bind_result($stmtu, $usn, $umwe, $upmi, $umd, $udc);
    }
}
if ($unum>0) { $gv=0;
?>
<p>&nbsp;</p>
<table class="table table-bordered table-hover table-striped table-condensed" style="width:75%;
margin:0px auto;" >
<tr>
<th>S/N</th>
<th>CANDIDATE NOMINAL MWE</th>
<th>PMI</th>
<th>MD</th>
<th>DC</th>
</tr>

<?php while (mysqli_stmt_fetch($stmtu)) { $gv++; ?>
<tr>
<td><?php echo($gv); ?></td>
<td><?php echo $umwe; ?></td>
<td><?php echo $upmi; ?></td>
<td><?php echo $umd; ?></td>
<td><?php echo $udc; ?></td>
</tr>
<?php } ?>

</table>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p><?php echo($bb); ?></p>

<?php
}
?>

```

Appendix 13: annotate1.php

```

<!doctype html>
<html>
<head>
<?php include(" head.php"); ?>
</head>
<body>
<?php include(" top.php"); ?>
<?php include(" dataconn.php"); ?>
<?php
$query="select dtext from dtext where sn=1 ";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        mysqli_stmt_bind_result($stmtu, $udtext);
        mysqli_stmt_fetch($stmtu);
    }
}

```



```



```

Appendix 14: processannotate.php

```

<!doctype html>
<html>
<head>
<?php include("head.php"); ?>
</head>
<body>
<?php include("top.php"); ?>
<?php include("dataconn.php"); ?>
<?php if(isset($_POST['annotate'])){
    $who = 'choice' . $_POST['who'];
?>
<div style="text-align:center;">
<h2>Thank you for your effort!</h2>
If you are done, simply close the page.<br>
However, if you want to go back and edit your decision, <a href="annotate<?php echo($_POST['who']);
?>.php">click here</a>.
</div>

<?php
$query="select sn from mwes";
$stmt=mysqli_stmt_init($csn);
$num=0;
if (mysqli_stmt_prepare($stmt,$query)) {
    if(mysqli_stmt_execute($stmt)){
        mysqli_stmt_store_result($stmt);
        $num = mysqli_stmt_num_rows($stmt);
        mysqli_stmt_bind_result($stmt, $sn);
    }
}
if ($num>0) { ?>

<?php while (mysqli_stmt_fetch($stmt)) {
    $nn = "de" . $sn;
    $what = $_POST[$nn];

    $sql="update mwes set ".$who."=? where sn = ?";
    $stmt=mysqli_stmt_init($csn);
    if (mysqli_stmt_prepare($stmt,$sql)) {
        mysqli_stmt_bind_param($stmt,"si",$what,$sn);
        if(!mysqli_stmt_execute($stmt)){
            $bb=$bb."<div>Error: Could not update $sn: " . mysqli_error($csn) . "</div>";
        } else {
            $bb=$bb."<div>$sn Successfully Updated!</div>";
        }
        mysqli_stmt_close($stmt);
    } else {

```

```

        $bb="<p>Error: Could not prepare query. " . mysqli_error($csn) . "</p>";
    }
} ?>

<?php
}
?>

<?php
$dinput=$_POST['missingones'];
$dinput2=$_POST['who'];
$pyscript = escapeshellcmd(' /home/nislgt5/public_html/mwe/000/pos600.py
');
$python = escapeshellcmd('/opt/imh-python/bin/python3.7');
$pydata = escapeshellarg($dinput);

exec(" $python $pyscript $dinput2 $pydata " , $result, $return);
?>

<?php
//Adding to cumulative lexicon, if selected by both annotators. The unique mwe attribue and ignore
parameter prevents duplicates as well as ensuring new additions of non-existent records.
$query="INSERT          ignore          INTO          lexicon
(mwe,archived,mwedes,compdes,combpas,mwemeanig,mwesample)          SELECT
mwe,archived,mwedes,compdes,combpas,mwemeanig,mwesample from mwes where choice1='Yes'
and choice2='Yes'";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
    }
}
?>

<?php
//Adding to cumulative NOT lexicon, if rejected by both annotators. The unique mwe attribue and ignore
parameter prevents duplicates as well as ensuring new additions of non-existent records.
$query="INSERT          ignore          INTO          not_lexicon
(mwe,archived,mwedes,compdes,combpas,mwemeanig,mwesample)          SELECT
mwe,archived,mwedes,compdes,combpas,mwemeanig,mwesample from mwes where choice1<>'Yes'
and choice2<>'Yes'";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
    }
}
?>

<?php
//delete from cumulative lexicon if an existing entry has been rejected by both annotators
#$query="DELETE FROM lexicon WHERE mwe in (SELECT DISTINCT mwe FROM not_lexicon)";

$query="DELETE FROM lexicon WHERE mwe in (SELECT DISTINCT mwe FROM mwes where
choice1<>'Yes' and choice2<>'Yes')";

$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){

```

```

        mysqli_stmt_store_result($stmtu);
    }
}
?>

<?php
//delete from cumulative NOT lexicon if an existing entry has been accepted by both annotators
#$query="DELETE FROM not_lexicon WHERE mwe in (SELECT DISTINCT mwe FROM lexicon)";

$query="DELETE FROM not_lexicon WHERE mwe in (SELECT DISTINCT mwe FROM mwes where
choice1='Yes' and choice2='Yes')";

$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
    }
}
?>

<?php
}
?>

<?php include(" bottom.php"); ?>
<?php include(" foot.php"); ?>
</body>
</html>

```

Appendix 15: seeresult.php

```

<!doctype html>
<html>
<head>
<?php include(" head.php"); ?>
</head>
<body>
<?php include(" top.php"); ?>
<?php include(" dataconn.php"); ?>
<?php
$query="select dtext from dtext where sn=1 ";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        mysqli_stmt_bind_result($stmtu, $udtext);
        mysqli_stmt_fetch($stmtu);
    }
}
?>
<div style="text-align:center; margin-bottom:50px; padding-bottom:20px; border-bottom:1px solid
#222;">
<h2>Original Text</h2>
<form id="form1" name="form1" method=" post" action="index.php" accept-charset=" utf-8">
<textarea name="contenter" id="editor1" cols="95" rows="10" disabled readonly><?php echo
$udtext; ?></textarea>
</form>
</div>
<div style=" background-color:#ccc; font-size:24px; padding:25px 50px; margin:0px 150px;">
<?php
$query="select count(*) as tp from mwes where archived='No' and choice1='Yes' and choice2='Yes'";

```

```

$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        mysqli_stmt_bind_result($stmtu, $tp);
        mysqli_stmt_fetch($stmtu);
    }
}
?>
<?php
$query="select count(*) as fp from mwes where archived='No' and (choice1<>'Yes' or
choice2<>'Yes')";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        mysqli_stmt_bind_result($stmtu, $fp);
        mysqli_stmt_fetch($stmtu);
    }
}
?>
<?php
$query="select count(*) as selected from mwes where archived='No' ";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        mysqli_stmt_bind_result($stmtu, $selected);
        mysqli_stmt_fetch($stmtu);
    }
}
?>
<?php
$query="select count(*) as fn from mwes where archived='Yes' and choice1='Yes' and choice2='Yes' ";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        mysqli_stmt_bind_result($stmtu, $fn);
        mysqli_stmt_fetch($stmtu);
    }
}
?>
<?php
$query="select count(*) as relevant from mwes where choice1='Yes' and choice2='Yes' ";
$stmtu=mysqli_stmt_init($csn);
if (mysqli_stmt_prepare($stmtu,$query)) {
    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        mysqli_stmt_bind_result($stmtu, $relevant);
        mysqli_stmt_fetch($stmtu);
    }
}
?>
<?php
$precision = $tp / $selected;
$recall = $tp / $relevant;
?>

<table width="100%">
<tr>

```

```
 <b>tp: </b><?php echo($tp); ?><br> <b>fp: </b><?php echo($fp); ?><br> <!--<b>Selected: </b><?php echo($selected); ?><br>--> <b>Precision: </b><?php echo(round($precision,2)); ?><br> </td>  <b>tp: </b><?php echo($tp); ?><br> <b>fn: </b><?php echo($fn); ?><br> <!--<b>Relevant </b><?php echo($relevant); ?><br>--> <b>Recall: </b><?php echo(round($recall,2)); ?><br> </td> <td> <?php $fp = $tp / ($tp + (0.5 * ($fp + $fn))); ?> <b>F<sub>1</sub> Score: </b><?php echo(round($fp,2)); ?><br> </td> </tr> </table> </div>  <div> <a href="index.php">Go back to Home</a> </div> <p>&nbsp;</p> <p>&nbsp;</p> <h2>Validated MWEs</h2> <?php $query="select sn,mwe,pmi,md,dc,archived,choice1,choice2,mwedec,compdes,combat from mwes where choice1='Yes' and choice2='Yes' order by mwe asc"; $stmtu=mysqli_stmt_init($conn); $num=0; if (mysqli_stmt_prepare($stmtu,$query)) {     //mysqli_stmt_bind_param($stmtu);      if(mysqli_stmt_execute($stmtu)){         mysqli_stmt_store_result($stmtu);         $num = mysqli_stmt_num_rows($stmtu);         mysqli_stmt_bind_result($stmtu, $sn, $mwe, $pmi, $md, $dc, $archived, $uchoice1, $uchoice2, $umwedec, $ucompdes, $ucombat);     } }  if ($num>0) { $gv=0; ?> <p>&nbsp;</p> <table class="table table-bordered table-hover table-striped table-condensed" style="width:100%; margin:0px auto;"> <tr> <th>S/N</th> <th>NOMINAL MWE</th> <th>PMI</th> <th>MD</th> <th>DC</th> <th>MWE Description</th> <th>Component Description</th> <th>Combination Pattern</th> <th>Initial Choice</th> <th>Annotator 1</th> <th>Annotator 2</th> | |
```

```

</tr>

<?php while (mysqli_stmt_fetch($stmtu)) { $gv++; ?>
<tr>
<td><?php echo($gv); ?></td>
<td><?php echo $mw; ?></td>
<td><?php echo $upmi; ?></td>
<td><?php echo $umd; ?></td>
<td><?php echo $udc; ?></td>
<td><?php echo $umwedes; ?></td>
<td><?php echo $ucompdes; ?></td>
<td><?php echo $ucombp; ?></td>
<td><?php if ($uarchived=="Yes"){ echo("No"); } else { echo("Yes"); } ?></td>
<td><?php echo $uchoice1; ?></td>
<td><?php echo $uchoice2; ?></td>
</tr>
<?php } ?>

</table>
<?php
}
?>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

<h2>Not Validated MWEs</h2>
<?php
$query="select sn,mwe,pmi,md,dc,archived,choice1,choice2,mwedes,compdes,combp from mwes
where choice1<>'Yes' or choice2<>'Yes' order by mwe asc";
$stmtu=mysqli_stmt_init($cn);
$num=0;
if (mysqli_stmt_prepare($stmtu,$query)) {
    //mysqli_stmt_bind_param($stmtu);

    if(mysqli_stmt_execute($stmtu)){
        mysqli_stmt_store_result($stmtu);
        $num = mysqli_stmt_num_rows($stmtu);
        mysqli_stmt_bind_result($stmtu, $sn, $mw, $pmi, $md, $dc, $uarchived,
        $uchoice1, $uchoice2, $umwedes, $ucompdes, $ucombp);
    }
}

if ($num>0) { $gv=0;
?>
<p>&nbsp;</p>
<table class="table table-bordered table-hover table-striped table-condensed" style="width:100%;
margin:0px auto;">
<tr>
<th>S/N</th>
<th>NOMINAL MWE</th>
<th>PMI</th>
<th>MD</th>
<th>DC</th>
<th>MWE Description</th>
<th>Component Description</th>
<th>Combination Pattern</th>
<th>Initial Choice</th>
<th>Annotator 1</th>

```

```

<th>Annotator 2</th>
</tr>

<?php while (mysqli_stmt_fetch($stmtu)) { $gv++; ?>
<tr>
<td><?php echo($gv); ?></td>
<td><?php echo $umwe; ?></td>
<td><?php echo $upmi; ?></td>
<td><?php echo $umd; ?></td>
<td><?php echo $udc; ?></td>
<td><?php echo $umwedes; ?></td>
<td><?php echo $ucompdes; ?></td>
<td><?php echo $ucommpat; ?></td>
<td><?php if ($sarchived=="Yes"){ echo("No"); } else { echo("Yes"); } ?></td>
<td><?php echo $uchoice1; ?></td>
<td><?php echo $uchoice2; ?></td>
</tr>
<?php } ?>

</table>
<?php
}
?>
<p>&nbsp;</p>
<p>&nbsp;</p>

<form id="form1" name="form1" method="post" action="processxml.php">
<div style="text-align:center;">
<input name="annotate" type="submit" id="annotate" value=" Process XML-LMF for Current Text "
/>
</div>
</form>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

<form id="form1" name="form1" method="post" action="processxmleumulative.php">
<div style="text-align:center;">
<input name="annotate" type="submit" id="annotate" value=" Process Cumulative XML-LMF. May
take a long time. " />
</div>
</form>
<p>&nbsp;</p>
<p>&nbsp;</p>

<?php include("bottom.php"); ?>
<?php include("foot.php"); ?>
</body>
</html>

```

Appendix 16: processxml.php

```

<!doctype html>
<html>
<head>
<?php include("head.php"); ?>
</head>
<body>
<?php include("top.php"); ?>

<h2>XML Processing</h2>

```

```

<?php
$python = escapeshellcmd('/opt/imh-python/bin/python3.7');
$pyscript = escapeshellcmd('/home/nisltg5/public_html/mwe/000/pos400.py');

$myfile = fopen($pyscript, "r") or die(" Unable to open file1!");
$bel = fread($myfile,filesize($pyscript));
fclose($myfile);

$aa = $python . " - <<'EOF'\n" . $bel . "\nEOF";
$output = exec($aa, $result, $return);

$g = implode(" ", $result);
echo($g);
?>

<hr>
<a href="XML_LMF.xml" target="_blank" download="XML_LMF.xml">Download XML File</a>
<p>&nbsp;</p>

<div>
<a href="index.php">Go back to Home</a>
</div>
<?php include("bottom.php"); ?>
<?php include("foot.php"); ?>
</body>
</html>

```

Appendix 17: processxmldcumulative.php

```

<!doctype html>
<html>
<head>
<?php include("head.php"); ?>
</head>
<body>
<?php include("top.php"); ?>
<h2>XML Processing (Cumulative)</h2>
<?php
$python = escapeshellcmd('/opt/imh-python/bin/python3.7');
$pyscript = escapeshellcmd('/home/nisltg5/public_html/mwe/000/pos500.py');

$myfile = fopen($pyscript, "r") or die(" Unable to open file1!");
$bel = fread($myfile,filesize($pyscript));
fclose($myfile);

$aa = $python . " - <<'EOF'\n" . $bel . "\nEOF";
$output = exec($aa, $result, $return);

$g = implode(" ", $result);
echo($g);
?>
<hr>
<a href="XML_LMF_Cumulative.xml" target="_blank"
download="XML_LMF_Cumulative.xml">Download XML File</a>
<p>&nbsp;</p>

<div>
<a href="index.php">Go back to Home</a>
</div>
<?php include("bottom.php"); ?>
<?php include("foot.php"); ?>

```

```
</body>  
</html>
```

Gabriel 1902228 Project Report

GRADEMARK REPORT

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

PAGE 92

PAGE 93

PAGE 94

PAGE 95

PAGE 96