Free Software Tools for Computational Linguistics: An Overview

Miloš D. Đurić
Faculty of Electrical Engineering, University of Belgrade, Belgrade, Serbia
djuric@etf.bg.ac.rs

Abstract: In the past six decades or so, we witnessed a rapid growth in the study of what is now known as Computational Linguistics. During the last decade, free software tools received increasing attention by the computational linguistic research community, and attracted the interest of computational linguists. This paper looks into the implications of utilizing free software tools in the domain of acoustic phonetics, discourse analysis and computational text analysis. In other words, the present paper is a descriptive exploration of free software tools, which I utilized for my research purposes. Therefore, this paper sets out to explore some useful aspects of these tools in order to get a better understanding of the roles they may have in computational linguistics. The aim of the paper is to provide a unitary descriptive account of Praat, KH Code and NLTK and shed light on their benefits from the point of view of their user. By way of demonstrating some of their features, certain concrete proposals are given. It is hoped that this investigation may spark interest for further research on the subject.

Keywords: Computational Linguistics; Free Software Tools; Praat; KH Coder; NLTK.

I. Introductory Remarks

Generally speaking, according to the pertinent literature, since their first appearance, in the late 1940s, computers have become increasingly familiar to the general public [1]. However, at the time, computational linguistics was seen mainly through mechanical translation, which was considered as the best known and most glamorous aspect of computational linguistics. With the advent of the Internet, Computational Linguistics has witnessed a revived interest precisely because it has become part and parcel of the phenomena Computer Science and Artificial Intelligence have set about to explain. The terminological inconsistency is also spotted in the pertinent literature, particularly since some researchers equate the terms ‘computer speech and language processing’ with ‘human language technology’, ‘natural language processing’, and ultimately, with ‘computational linguistics’ [2].

Broadly speaking, computational linguistics is said to be interdisciplinary, since in its methods of analysis it takes into account a variety of diverse perspectives. More specifically, in linguistics this area has the implications and applications in the domain of Second Language Acquisition and Computer-Assisted Language Learning [3], and, it can be also applied in the domain viewed from information-processing perspective in works which treat humans as limited-capacity processors. As regards the theoretical framework, which was fused in an eclectic way in order to analyse the corpus-based data, my analysis has been informed by the following studies: [4]–[26].

The paper aims to describe three free software tools in terms of their usability for a linguist, and a computational linguist, for that matter, and is, therefore, primarily descriptive in its orientation. The practical part of the study is devoted to bringing together linguistically and computationally motivated analyses as a rationale behind the inspection of free software tools for computational linguistics.

II. Free Software Tools

Before one proceeds, one is tempted to provide at least one broad and tentative working definition of free software. Broadly speaking, free and open-source software (or, simply FOSS) might be defined as a piece of software that can be classified as both free software and open-source software. According to the pertinent literature, the terms “free software” and “open source software” might be said to refer to software products distributed under terms, which allow users to use the respective software, modify the software according to one’s needs, and ultimately, redistribute the software [27].

Since my intention is not to clarify this delimitation in depth, I shall adopt solely this working definition and apply it to the tools that were utilised in the research, and this survey, for that matter.

Equally, the term “tools” will be used very loosely, since one encompasses an acoustic tool (Praat), a text chunking tool (KH Coder), and a powerful library based on the Python programming language (NLTK), respectively.

A. Praat

Broadly speaking, Praat is a free software package with open source code aimed at linguists intending to analyse speech, i.e. spoken discourse. In addition to this, according to the pertinent literature, Praat is generally defined as computer software for phonetic analysis [28], and more specifically, as a standard tool for transcription of speech, and classification of speech events [29]. In addition to these definitions, one comes across the definition of Praat being described as a versatile, open-source platform, which provides a whole lot of features. Furthermore, the quoted reference [28] asserts that Praat might be utilised in the context of the pronunciation teaching process by allowing the learners to individually analyse the generated visual patterns of their own speech thereby making them aware of nuances and diverse

distinctions within the target language pronunciation. In addition to this, the research has been undertaken in order to explore the ways in which the learners of foreign languages could improve their pronunciation by using Praat.

Chronologically speaking, Praat is also defined as an application developed for speech researchers. The creators of Praat are Paul Boersma and David Weenik (both from the University of Amsterdam). Even though the main purpose of Praat was to apply it in the realm of speech analysis and speech synthesis, its application has been developed further in the direction of facilitating manipulation and labelling processes whilst, simultaneously, offering a powerful apparatus for phoneme identification. Additionally, Praat researchers enhanced format plotting, amongst other things, thereby providing a sound foundation for teaching vowel and diphthong production processes.

It goes without saying that the Praat program can be downloaded free of charge [30]. The Internet source also provides the description of the features pertaining to the Praat software tool as well as useful guides.

However, Praat is not only used in the context of Second Language Acquisition (i.e. SLA), but also in the context of prosody conversion [31]. Praat seems to allow for all sorts of articulatory and acoustic analyses. These analyses comprise segmental and prosodic characteristics of spoken discourse.

According to some researchers, Praat might tackle dialect research and may even be used for forensic purposes, since it already enables a detailed acoustic analysis and annotation of speech data, both in phonetic and phonological domains [32]. Voice analysis using Praat tool has also been fruitful so far, particularly in the domain of assessing a user's emotional state [33].

The Praat research is anchored in different theories, one of which is Optimality Theory (OT), particularly as a way of understanding the Optimality-theoretic driven stochastic grammars [34]. A particularly striking example of the application of OT theory in the Praat analysis environment lies in Boersma's Gradual Learning Algorithm enabled by the Praat program to help you rank Optimality-Theoretic constraints in ordinal and stochastic grammars.

In the domain of language teaching, Praat is considered to have been designed to be used by serious speech researchers, whilst complex computer readouts related to formant plots demand a sophisticated level of understanding [35]. In teaching English pronunciation practice, the focus is primarily on segmental and suprasegmental pronunciation [36]. Nevertheless, Praat is also utilised in looking into its effectiveness in helping students to acquire prosodic features of the English language [37].

Before I embark on the concrete application, let us see the plausible application, guaranteed by the creators of Praat. Firstly, Praat can be used in speech analysis, which is comprised of spectral analysis, pitch analysis, formant analysis, intensity analysis, analysis of jitter, shimmer and voice breaks, generating cochleagram and excitation pattern. Secondly, it is used in the domain of speech synthesis, which brings into the focus pitch, formant, intensity, articulatory synthesis, as well as Klatt acoustic synthesis. Praat also marks the borderline in the domain of listening experiments and labelling and segmentation. The former comprises identification and discrimination tests, whilst the latter includes label intervals and time points on multiple tiers, the use of phonetic alphabet, and the use of sound files up to two gigabytes 2 GB, or in terms of corpus length three hours of spoken data.

One ought to mention other functionalities of Praat for the purpose of a more comprehensive picture. These functionalities would include: 1. speech manipulation (encompassing: change pitch, duration contours and filtering), 2. learning algorithms (bringing about a biologically-inspired feed forward neural networks, followed by discrete and stochastic Optimality Theory), 3. statistics couched in multidimensional scaling, principal component analysis and discriminant analysis, 4. graphics (high quality for scientific papers and theses, production of encapsulated PostScript files, integrated mathematical and phonetic symbols), 5. programmability (easy programmable scripting language and well-established communication with other programs), 6. Portability (including well-organised machine-independent binary files, and possibility of reading and writing diverse sound and other file types, and finally, 7. configurability [38]. It should be mentioned that Praat abounds in plug-ins, which are resorted to in prosody analysis [39], amongst other things.

Now let us see the screen capture of the Praat working environment.

It can be spotted in Figure 1, that the Praat working environment comprises two principal elements: the Praat Objects, and the Praat Picture. According to the pertinent literature, the Praat Objects window is the location for the majority of workflows, and this menu is used to open, create and save files, with further possibility of opening various editors and queries one needs in order to work with sound files [40]. One should select a sound and then the option "View and Edit". Afterwards, whilst examining a sound file, the editor window shows the sound's waveform on the top and a spectrogram on the bottom. Within this working environment, the cursor

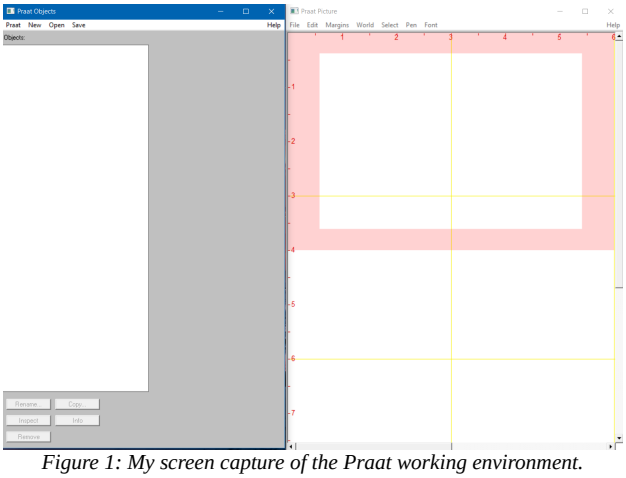

Figure 1: My screen capture of the Praat working environment.

allows a researcher to carry out selections and perform measurement. Generally speaking, Praat is particularly useful in corpus-based analysis. A spoken corpus typically consists of a set of sound files, each of which is paired with an annotation file, and metadata information. In the part that follows, I shall describe the application of Praat in my research.

The observations I make in this investigation are based on the data that have been collected from the oral medium in the form of academic lectures. Namely, academic discourse manifests a wealth in the number and variety of compounds. Generally speaking, delimiting binary/two-constituent/non-canonical compounds has not always been fairly easy and not without problems. Since stress is considered to be one of the reliable criteria (for example, see [41]–[45]), it was interesting to look into the role of stress in compounds by means of acoustic analysis provided by Praat. However, generally speaking, non-canonical i.e. multi-constituent compounds have been out of the focus due to certain delimitation problems, among other things. If they have been discussed at all, this mainly occurred in connection with standard language and written medium, as well as fairly informal styles. It seems that multi-constituent constructs in academic discourse have been left aside. As a consequence of such tendencies, multi-constituent constructs have been delimited as a separate, though not syntactically clearly delimited category of lexical items. Strictly speaking, this Praat-motivated investigation turns attention to the issue of a more adequate delimitation of multi-constituent constructs, particularly to the set of linguistic units that display variation in stress, this being illustrated by the corpus-based data.

The motivation lying behind the decision to select compounds as an object of study could be found in the claims from the pertinent literature, according to which, the analysis of nominal compound constructions has proven to be an unmanageable and recalcitrant problem, which poses serious challenges for natural language processing systems [46].

More precisely, in this Praat-motivated study, I focus on stress of multi-constituent constructs. Since stress is often used as the delimitation marker between phrases and compounds in the English language (see, for example: [47]–[49]), and yet, many examples taken from the language contradict this general rule, I have decided to analyse the authentic oral corpus, and to analyse how multi-constituent constructs behave in this discourse type with respect to this prosodic feature.

Our speech data come from a specific register of the oral/spoken medium in the form of academic lectures. I have chosen this type of spoken discourse because I have assumed that there could be either consistency or variability in the prosodic pattern of certain multi-constituent constructs, which are used relatively frequently in academic discourse. Furthermore, this discourse type provides a relatively narrow domain of knowledge in which such constructs are used. The examples that I considered relevant for my hypothesis showed that relevant factors for the occurrence of compound stress consistency might be the processes of domain-specific lexicalization of certain constructs.

In order to avoid the mentioned problems, I extended the empirical scope.

The question that might be posed is: Why Computer Science academic discourse? The first reason, according to the pertinent literature would be that from the perspective of the traditional lexicon designer working within computational linguistics, complex nominals, i.e. compounds are formed generatively and therefore do not merit explicit listing except when clearly non-compositional [50]. In this context, according to the quoted reference, in this spectrum of compounds, technical terminology holds the attention of a significant location, being highly productive and encapsulating the essential concepts of a particular technical domain.

The second reason for selecting five academic lectures was that I wanted to avoid using fairly small data sets. The third reason pertains to my wish to avoid using my own intuition, the practice not uncommon in the linguistic research of researchers operating within the tradition of transformational-generative paradigm and transformational grammarians who have long used their own native speaker intuitions [51].

I have extended the empirical scope and studied the prominence found in the actual speech (i.e. speech data from more specialised genres and language registers), and tried to analyse these linguistic items by means of acoustic analysis. I have established five classes of constructs: 1. Dictionary-attested constructs (DAC), 2. Frequent and repeated constructs (FRC), 3. Discourse community constructs (DCC), 4. Domain-specific constructs (DSC) and 5. Multi-constituent constructs (MCC).

Multi-constituent constructs (MCCs) were selected for the analysis. My initial assumption is that there could be either consistency or variability in the prosodic pattern of MCCs in academic discourse. Corpus comprises high-quality recordings of lectures (the duration of which totals six hours and fourteen minutes in the MP3 format which was subsequently converted into .wav format so as to be able to undergo the Praat analysis. The Transcript of Lectures (ToL) consists of 75 pages comprising 45187 words.

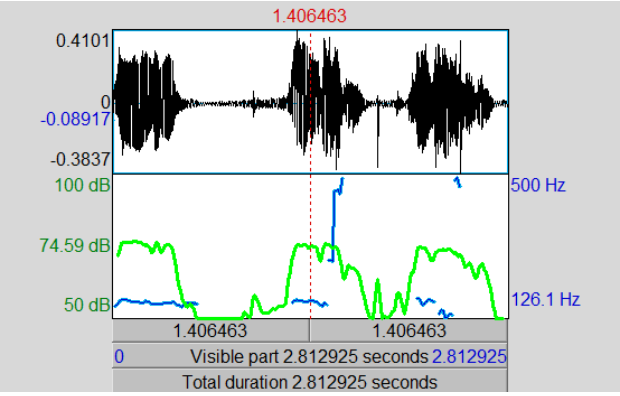

Figure 2: The Praat-generated token 1 of the MCC "random number generator" from my corpus.

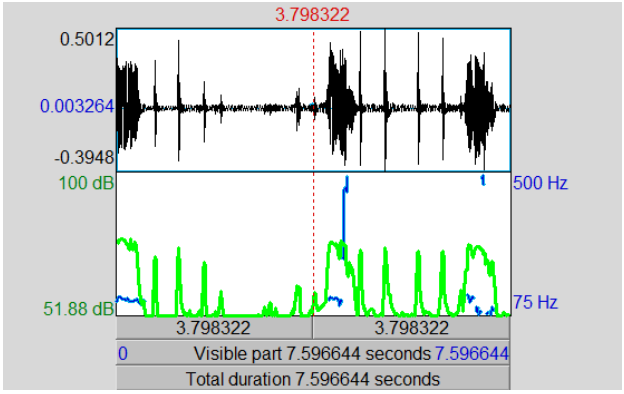

Figure 3: The Praat-generated token 2 of the MCC "random number generator" from my corpus.

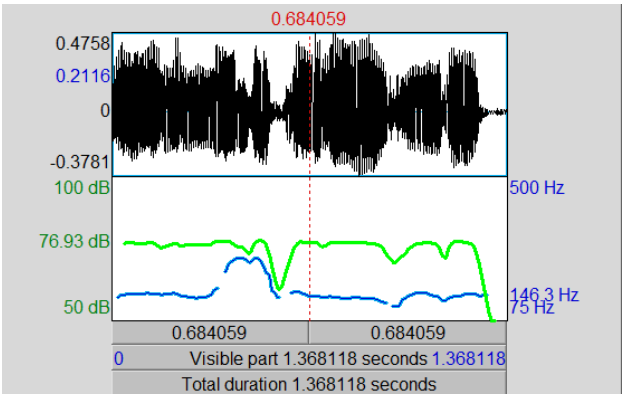

Figure 4: The Praat-generated token 3 of the MCC "random number generator" from my corpus.

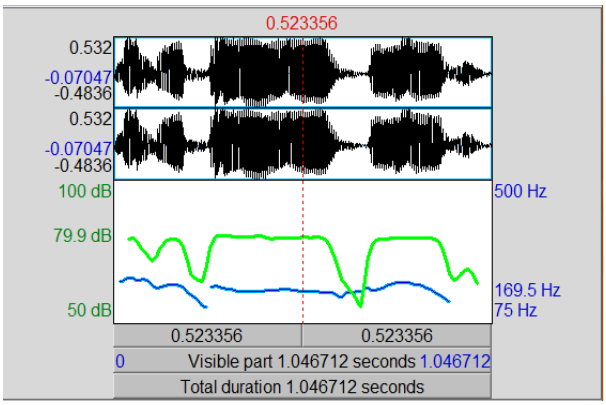

Figure 5: The Praat-generated token 1 of the MCC "hundred dollar bills" from my corpus.

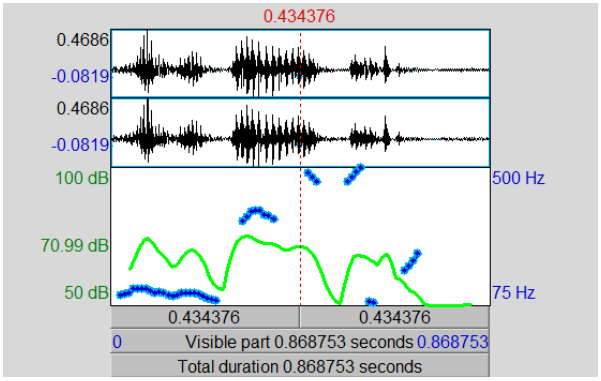

Figure 6: The Praat-generated token 2 of the MCC "hundred dollar bills" from my corpus.

Set apart from mostly clear-cut cases of binary compounds is a group of multi-constituent compounds which may exhibit somewhat different acoustic behaviour. Let us see the following example from our corpus. Specifically, I there are three tokens of the MCC "random number generator" in my corpus. All tokens have been analysed by means of Praat, and the results of the analysis are displayed in the following figures.

The maximum pitch for the first token equals 489.71 Hz (Figure 2), whilst for the second token this value equals 488.74 Hz (Figure 3), and 258.47 Hz in the case of the third token (Figure 4). The duration, displayed in seconds, varies, so, on the one hand the duration of the first token totals 2,81 s, whilst the second token lasts for 7,59 s and the third one 1,36 s. Linguists are feeling their way on a slippery terrain in the cases, like this one, when the intra-speaker variation has been spotted and then acoustically-confirmed. So, in a nutshell, the Praat-provided visualisation facilitates better understanding of the subtle differences in intra-speaker variation, otherwise perceived by introspection, but not confirmed acoustically by proper measurement. Let us now see the case of the MCC that has two tokens in total. The example in question is the unit "hundred dollar bills".

When juxtaposed, as in the case of this Praat-generated visualisation (Figure 5 and Figure 6), one cannot but notice that these two tokens of the MCC "hundred dollar bills" exhibit the variation in the maximum pitch. More specifically, the maximum pitch of the first token equals 208.16 Hz, whilst the latter one equals 493.80 Hz. Additionally, the variation in duration has been spotted. Namely, the first token lasts for 1.04 s, and the second token lasts for 0.86 s.

In my previous research, all the analysed examples have shown that compoundhood of a MCC is well-established in discourse unless for some discoursal reason the significance of the construct is to be underlined, for example, at the end of the sentence, or at the end of the discoursal subtopic in generalised conclusive utterances. This conclusion is enabled through the analysis by means of Praat. In the next section, we shall see some plausible advantages and disadvantages of using Praat.

In this part, I have tried to show how some central acoustic parameters provided by Praat can be applied in Computational Linguistics by focusing on a small group of compounds (i.e. MCCs) that might mark the borderline between binary i.e. canonical and multi-constituent i.e. non-canonical compounds. I have argued that the Praat visualisation and Praat-generated parameters could change the fairly static picture provided by non-acoustic approaches. Moreover, the non-acoustic analysis seems to be inadequate to grapple with items that cannot be easily captured in compoundhood-driven terms. In addition to this, intuition-based analyses of MCCs have equally brought about a host of problems, which can be resolved by means of pretty straightforward visualisations, such as those generated by means of Praat.

Perhaps the paramount feature of Praat might be considered to be its all-embracing help function, which is brought up-to-date regularly. It should be stressed that

this represents a circumstance which seems convenient for both expert and non-expert users. Over and above, another eye-catching and attractive Praat-feature refers to its offering of its own scripting language, which is another reason to utilise this tool in computational linguistics.

However, this software tool is not without its problems. More precisely, the felicitous handling and usage depends on the user. Namely, computational linguists might find this free software tool very useful and handy, whilst perhaps some non-acoustically oriented users with the lack of knowledge in the domain of acoustic phonetics and computational linguistics, for that matter, might find the utilisation of Praat as something pretty complex and demanding thereby opting for the more intuitive approach in spoken language data analysis.

To conclude, intuitive analyses based on sort of introspection have tended to obfuscate rather than clarify speech sound phenomena and suprasegmental properties of sounds, MCCs and their stress not being an exception to this problem set. The described free software tool Praat, which is intended for acoustic analysis, seems to offer both precise measurement and accurate description of the given speech phenomena under investigation.

The successful utilisation of this free software tool depends on the nature of the user. Namely, those users who are not familiar with concepts within computational linguistics will perhaps use Praat less successfully than those who are computational linguists. However, they are not without problems, particularly if we take into account computer scientists, software developers and engineers who can master this free software tool to overcome any acoustic problem. Therefore, it can be said that Praat depends on the nature of its user.

B. KH Coder

If one tries to define this software tool, one comes across the definition of KH Coder provided by its author. I have slightly modified the given definition by adding the item “tool” in the description. Namely, KH Coder is usually defined as a free software tool for quantitative content analysis or text mining, and it is also utilised for computational linguistics [52]. Furthermore, it is also characterised as a software tool intended for computer-assisted qualitative data analysis. KH Coder was developed by Koichi Higuchi.

The survey of the literature shows that KH Coder is successfully implemented in diverse text analyses, such as the analysis of occupational accidents and their prevention in Spanish digital press [53]. It is also used in analysing students’ course evaluation through text mining, which is predominantly based on co-occurrence network analysis provided by KH Coder [54]. KH Coder is also used in the context of SLA and EFL and ESL in preparing specific teaching materials for advanced reading comprehension based on specific text mining [55]. Furthermore, this free software tool is highly suitable for specific tasks, such as the analysis of specific keywords with the help of co-word mapping comparison between two types of newspapers [56].

Certain authors explore the big data realm as a completely novel field for both scholars and practitioners dealing with big data conceptualisation based on diverse case studies [57]. The relevant features of KH Coder, such as multi-dimensional scaling, cluster analysis and co-occurrence network, are employed by researchers whose aspirations are to be found in the domain of specific language register. In this sense, the researchers implement KH Coder in order to carry out multi-dimensional scaling and co-occurrence network analysis on the academic journal dataset [58].

The emerging field of quantitative text analysis also represents a fruitful field of research particularly for the authors utilising the given free software tool, which has proven to be a satisfactory testing ground both for written and oral data [59]. Similarly, KH Coder is used by researchers exploring news articles databases and comparing their local and international media reports [60]. Finally, there are authors who employ text analytics visualisation provided by the free software tool in question in order to explore and visualise student comment data in the discourse of science and technology [61].

In the part that follows I shall briefly describe one previous research of mine, in which I utilised KH Coder for computational discourse analysis [62]. The research was part of a wider interdisciplinary field of discourse studies, more specifically, digital art museum discourse, which explored various aspects of language expression that is manifested in this discourse type. CAT was applied to the text contained within the web pages of six digital museums of digital art. I must emphasise that I have utilised some parameters, or, more specifically, textual dimensions elaborated in the pertinent literature (for instance, see [63]). At this point, one should also add the remark from the literature that text collections and corpora in digital form (like my corpus) represent important resources for empirical research [64].

Since KH Coder belongs to free software tools for quantitative content analysis and text mining, it is, consequently utilised for computational linguistics, and as such offers a plethora of features that might analyse the language material and facilitate CTA. By way of illustration, we shall see the actual implementation of this software tool.

The given visualisation (Figure 7), provided by KH Coder, lends support to the assumption that individual language items might be followed easily, even though a lot of combinations would appear within these clusters. Additionally, some overlapping clusters might have gone further on the analysis path leaving the most distant ones stranded. One can notice that the lexical unit “programming” collocates with the units “software”, “code” and “package” thereby generating the following clusters: “programming software”, “programming code” and “programming package”, to mention but a few.

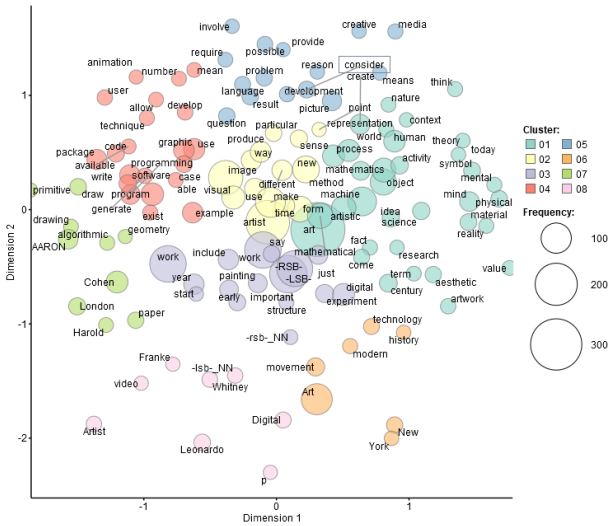

Figure 7: The two-dimensional solution for non-metric multidimensional scaling (2D Cruscal) for the text excerpt from my DAM corpus.

Now, let us see the three-dimensional solution.

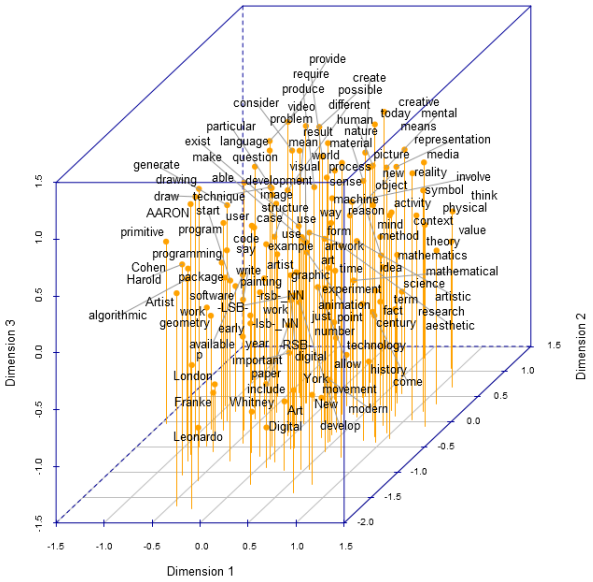

Figure 8: The three-dimensional solution for non-metric multidimensional scaling (3D Cruscal) for the text excerpt from my DAM corpus.

The first impression is that 3D cruscal (Figure 8) seems not to be neatly organised as is the case with the 2D cruscal. Perhaps, this might be the case due to the corpus size. However, 3D visualisation seems to offer less satisfactory data when it comes to cluster analysis. Nonetheless, a host of collocations can be traced and spotted without looking into separate tables, for that matter.

Now let us see the visualisation of the previously sifted corpus data.

The lexical unit “VR” collocates with the items “people”, “platform”, “time”, etc. And the adjective “virtual” generates the clusters “virtual museum”, “virtual exhibition” and “virtual experience” (Figure 9).

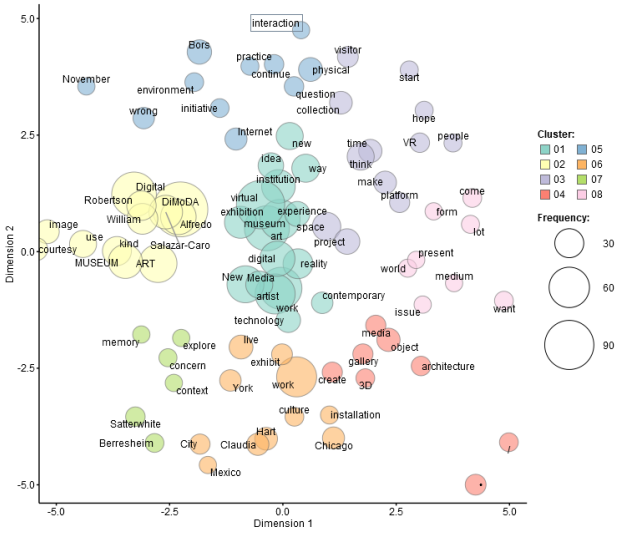

Figure 9: 2D Cruscal for the text excerpt from my DiMoDA corpus.

And now let us see another case of the given corpus-based analysis.

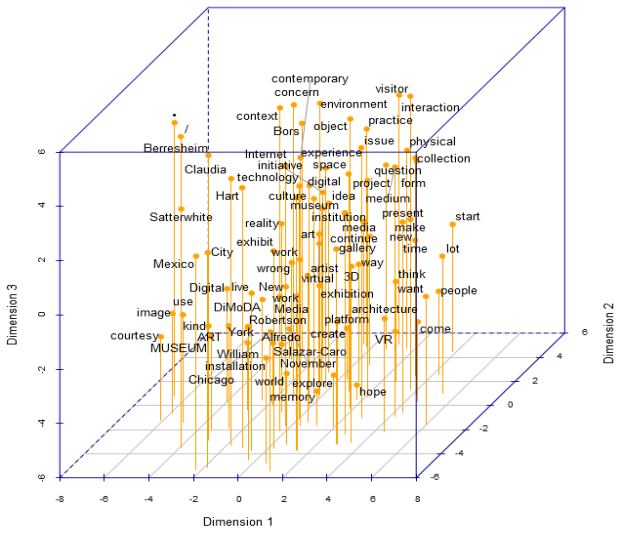

Figure 10: 3D Cruscal for the text excerpt from my DiMoDA corpus.

This time (Figure 10), 3D cruscal fits neatly in the representational-computational approach to the analysed lexical items. The centrality is taken by the lexical item “museum” which is located near items with which it enters into the most frequently occurring collocation patterns. Of course, this visualisation is not sufficient enough on its own, but ought to be accompanied by statistical tables and other numerical parameters that are obtainable in KH Coder.

Sometimes automatically-driven part-of-speech tagging might be problematic, as can be seen in the previous illustrative example (Table 1). Namely, the semantic unit “New” is treated as an instance of a proper noun, even though we cannot see the context in which it appears immediately and the sole indicator for this

Table 1: The KH Coder-generated illustrative table for the text excerpt from my La TurboAvedon corpus.

Lexical items	Part of Speech	Frequency
AVEDON	ProperNoun	21
space	Noun	17
LATURBO	ProperNoun	16
work	Noun	16
virtual	Adj	10
New	ProperNoun	8
artist	Noun	8
live	Verb	8
avatar	Noun	7
consider	Verb	7
experience	Noun	7
media	Noun	7
paraspace	Noun	7
production	Noun	7
sculpture	Noun	7
surface	Noun	6
Sculpt	ProperNoun	5
authorship	Noun	5
identity	Noun	5
object	Noun	5
parasubject	Noun	5
polygon	Noun	5
social	Adj	5
term	Noun	5

decision seems to be the initial capital letter “N”, which must have evoked the item “New” in the proper noun“New York”. This is the reason why one should use simultaneously statistical tables and visualisations provided by KH Coder. Therefore, classical tables should be interpreted jointly with graphs and 2D and 3D visualisations. It is in this way that the computational discourse analysis might be extended.

KH Coder enables analysing lexical clusters and collocations that are not quite susceptible to classical text analysis, or discourse analysis, for that matter. However, this computational discourse analysis is in stark opposition to non-computational accounts, precisely because it provides better language data manipulation through precisely adjusted measurement methods in terms of statistical analysis.

As can be seen from the neat examples from my corpus cited so far, sometimes certain clusters signal to the analyser to drop certain definitional characteristics, as was the case with letter capitalisation influencing the specific part-of-speech tagging (see: Table 1). The analysis restriction that seems problematic refers to the situation when certain lexical units may remain unspecified in terms of part-of-speech. This is why one should not combine Chomskyan (i.e. computational) manners of analysis with non-Chomskyan (i.e. impressionistic) ways of analysis. It should be added that, according to the literature, in the period before Chomsky, linguistics tended to be a taxonomic enterprise, which was dubbed verbal botany [65]. However, this is not to say that KH Coder lacks the essential features of a free software tool intended for computational discourse analysis. On the contrary, this tool provides an upgrade of a sort. Following the standard picture, the benefits of KH Coder refer to the analyses it provides: 1. word frequency list, 2. the context in which the lexical item is used, 3. co-occurrence network of words, 4. correspondence analysis of words, to name just a few. However, some challenges remain unresolved, such as those referred to in the literature concerning the pitfalls on the path to formulating a unique query which could extract information from aligned texts [66], among other things.

Furthermore, statistical analyses of automatically extracted words are suitable for gaining a whole picture of the data since traditional problems of how to represent lexical items and their respective lexical clusters and collocations in standard glossaries and general dictionaries alike have become more apparent than real. Unfortunately, various coding rules to count concepts, a topic no less interesting in the computational discourse analysis could not be taken up in my investigation, and this paper, for that matter. In the part that follows, I present another free software tool.

C. NLTK

The Natural Language Toolkit (NLTK) represents a collection of libraries and programs for symbolic and statistical NLP written in the Python programming language. More precisely, according to the pertinent literature, the NLTK is a suite of open source program modules, tutorials and problem sets, providing ready-to-use computational linguistics courseware [67]. Furthermore, NLTK is said to be a platform for building Python programs to work with human language data. Additionally, it is asserted in the literature that Python Natural Language Processing Toolkit plays an important role as a platform for building Python programs to work with human language data [68].

NLTK arrives with a large collection of corpora, followed by large-scale and extended documentation, making NLTK unique in providing a comprehensive framework for students to develop a computational understanding of language [69]. The quoted reference asserts that NLTK’s code base of 100,000 lines of Python code includes support for corpus access, tokenising, stemming, tagging, chunking, parsing, clustering, classification, language modeling, semantic interpretation, unification, among other things.

Furthermore, NLTK has many third-party extensions. There are plenty of approaches to each NLP task in the NLTK environment. Related to this are also fast sentence tokenisation, and other relevant features for computational linguistics. And now let us consider the choice for selecting the Python programming language.

It should be mentioned that the creators of NLTK are Steven Bird and Edward Loper (both from the Department of Computer and Information Science at the University of Pennsylvania). NLTK has been used successfully as a teaching tool and, so far, many researchers have chosen Python as their implementation language for NLTK, mainly because Python’s syntax and semantics are transparent with good string-handling functionality. On the one hand, Python is an interpreted language which makes it suitable for facilitating interactive exploration. On the other hand, it is an object-oriented language, which entails that Python allows data

and methods to be encapsulated and readily and easily re-used.

Additionally, according to the literature, Python is heavily used in the industrial context and scientific research alike. Nevertheless, it also offers programming possibilities in educational contexts around the world. The same source claims that Python is said to be often praised for the way it facilitates productivity, quality, and maintainability of software [70].

Some of the features that might be useful to computational linguists can be carried out by means of the NLTK. For example, tokenising text into sentences, tokenising sentences into words, tokenising sentences using regular expressions, filtering stop words in a tokenised sentence, stemming words, lemmatizing words, creating custom corpora, part-of-speech tagging, extracting chunks, text classification and parsing specific data, to list just a small portion of features, functionalities and possibilities from the representative literature [71]. In the following section, I shall briefly describe the actual use of NLTK in CTA.

The first obvious application of NLTK in CTA refers to the main features pertaining to computing with language. More specifically, NLTK enables the following: categorising and tagging words, processing raw text, accessing text corpora and lexical resources, writing structured programs, learning how to classify a text, extracting specific text information from text. Moreover, one can also analyse the sentence structure by NLTK. This tool can also analyse the meaning of the language data in general, and the meaning of sentences, in particular.

The already existing corpora may be sufficient for a scholar interested in the basic computational linguistics. Still, they seem to be representative enough in terms of corpus representativeness. Let us see an example of NLKT corpora.

These nine texts (Figure 11), or more precisely, corpora are sufficiently equipped so as to serve as input data for a computational linguistic analysis. A linguist is provided with neatly modified ways of exploring the given corpora. For example, let us search for concordances of the lexical item “lucky” in the first corpus (Figure 12).

Counting vocabulary is another convenience provided in the NLTK environment. Let us see an example in Figure 13.

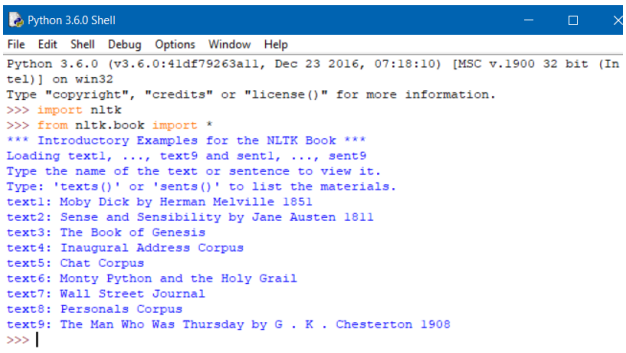

Figure 11: My screen capture of an illustrative example of the NLTK corpus structure.

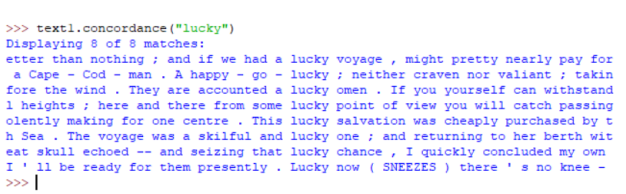

Figure 12: My screen capture of the NLTK-generated concordance of the lexical item “lucky” from the first NLTK corpus.

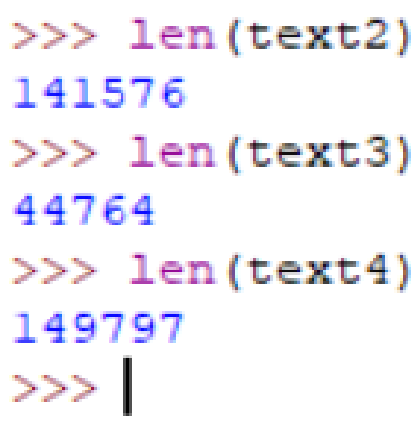

Figure 13: My screen capture of an illustrative example of vocabulary counting of NLTK corpora.

As seen from my illustrative example (Figure 13), the described free software tool seems to be very user-friendly and convenient for a computational linguist who wishes to find out the length of a corpus. Strictly speaking, the number refers to the words and punctuation symbols which occur. The term len is utilised to obtain the length of something, in my case, a text, which has been applied to the corpora at hand.

I was particularly interested in generating tokens and tokenisation process within the NLTK context. Not surprisingly, tokens have become one of the highly explored language phenomena within the current linguistic research both of cognitive and computational provenance (see, for instance, [72] and [73]). In the vast literature existing today, a token has been referred to as an instance of a unit, as distinct from the unit that is instanced [74]. More specifically, in linguistics, the term “token” is simply defined as a particular example of a general type [75]. According to some computationally-

motivated sources, however, the vocabulary of a text is just the *set* of tokens that it uses, since in a set, all duplicates are collapsed together [76]. Now, let us see a typical example of tokenisation by means of NLTK.

The immediate problem that is noticed (Figure 14) is the presence of the occurrence of orthographic symbols,

```
>>> sorted(set(text3))
['!', "'", '(', ')', ',', ',)', '.', '.)', ':', ';', ';)', '?', '?)', 'A', 'Abel
', 'Abelmizraim', 'Abidah', 'Abide', 'Abimael', 'Abimelech', 'Abr', 'Abrah', 'Ab
raham', 'Abram', 'Accad', 'Achbor', 'Adah', 'Adam', 'Adbeel', 'Admah', 'Adullami
te', 'After', 'Aholibamah', 'Ahuzzath', 'Ajah', 'Akan', 'All', 'Allonbachuth', '
Almighty', 'Almodad', 'Also', 'Alvah', 'Alvan', 'Am', 'Amal', 'Amalek', 'Amaleki
tes', 'Ammon', 'Amorite', 'Amorites', 'Amraphel', 'An', 'Anah', 'Anamim', 'And',
 'Aner', 'Angel', 'Appoint', 'Aram', 'Aran', 'Ararat', 'Arbah', 'Ard', 'Are', 'A
reli', 'Arioch', 'Arise', 'Arkite', 'Arodi', 'Arphaxad', 'Art', 'Arvadite', 'As'
, 'Asenath', 'Ashbel', 'Asher', 'Ashkenaz', 'Ashteroth', 'Ask', 'Asshur', 'Asshu
```

Figure 14: My screen capture of an illustrative specimen of tokens in the NLTK corpus number 3.

```
>>> text3.dispersion_plot(["light", "dark", "love", "Lord", "sin"])
```

Figure 15: My screen capture of an illustrative example written in Python in order to obtain the lexical dispersion plot for NLTK corpus 3 (i.e. The Book of Genesis).

which are redundant for a linguistic analysis, at least the one that is lexeme-focused. According to the literature on the subject, by wrapping `sorted()` around the Python expression `set(text3)`, a sorted list of vocabulary items is obtained, and this list begins with various punctuation symbols and continues with words starting with “A”. It is essential that we should mention that all upper-case words precede lower-case words.

However, whilst working within the NLTK environment, one may also determine the location of a lexical item in the given corpus. This positional information can be displayed by means of a lexical dispersion plot. Each stripe represents an instance of an item, and each row represents the entire corpus. Let us consider the following example.

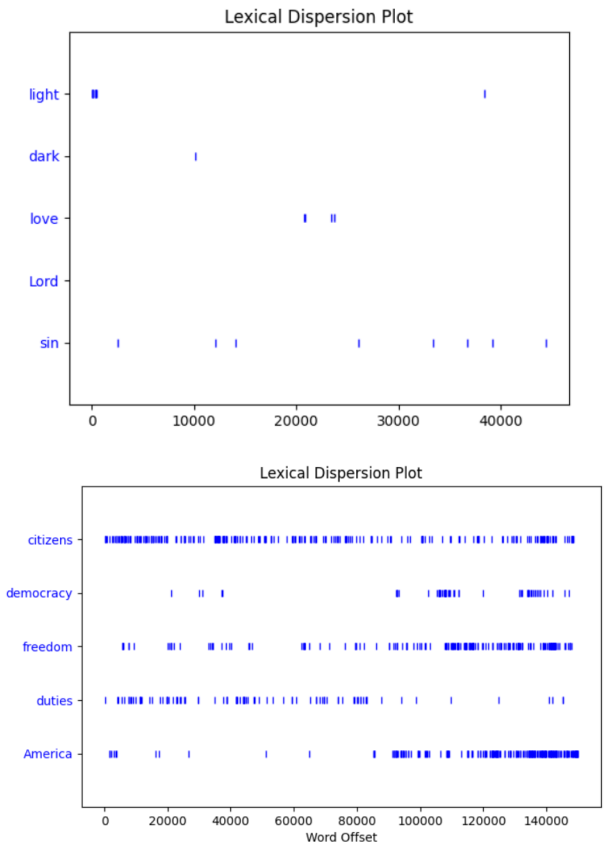

Figure 17: My screen capture of the lexical dispersion plot for the NLTK corpus 4 (i.e. Inaugural Address Corpus) generated by the NLTK tool.

```
>>>
>>> text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America"])
```

Figure 18: My screen capture of the command line written in Python in order to obtain the lexical dispersion plot for the NLTK corpus 4 (i.e. Inaugural Address Corpus).

My illustrative examples (Figure 15 and Figure 16) capture the gist of the given lexical dispersion. However, in the pertinent literature, there are examples which perhaps illustrate the point in a more visually striking way [70]. Let us now consider them in turn in Figure 18 and Figure 17, respectively.

Other useful features pertain to parsing and part-of-speech tagging. In linguistics, one cannot avoid grappling with the notion from the traditional grammar, which treats parsing as the pedagogical exercise of labelling the grammatical elements of single sentences [77]. It is claimed that the criteria of analysis leading to the identification of grammatical elements seem to be more salient mainly owing to the ways in which speakers of a natural language (i.e. English, in this case) use these items to relate sentences in the language in its entirety. When it comes to part of speech, one encounters the fuzzy traditional notion, which refers to a grammatical class of words [77]. Due to their inexplicitness when defined, and the restricted nature of their definitions, some linguists tend to prefer alternative terms (for example, in the pertinent literature, the term 'parts of speech' is equated with the terms: 'word classes' [78], 'class of words' [79], and 'lexical category' [80], to mention but a few labels, among other things). However, since the terminological inconsistency is not the object of this study, I shall not broaden further that, no less interesting, topic.

On the one hand, one may spot the linguistically-oriented parsing task, which is mainly the task of assigning words to parts of speech. However, on the other hand, in the computationally-oriented analyses, this parsing process refers more to the assignment of syntactic structures to sentences, especially by parsing programmes, or, more precisely, parsers [74]. From a cognitively oriented perspective, however, parsing is treated as one of the mental processes involved in sentence comprehension, in which the listener determines the syntactic categories of words, joins them up in a tree, and ultimately, identifies the essential parts of that particular sentence [81]. Now, let us consider the basic use of tagger in computational linguistics.

A part-of-speech tagger, or POS-tagger, processes a sequence of lexical items in a sentence from my corpus (“Andrew Telfer is writing a note at his desk in one corner of a big book-lined room.”), and attaches a part of speech tag to each item (e.g. ‘Andrew’, ‘NNP’, ‘Telfer’, ‘NNP’, and so on). Briefly, my example (Figure 19) illustrates how the implementation of the NLTK tool may be pretty convenient for a computational linguist wishing either to practice automatic tagging or to import a larger text portion in order to carry out a CTA.

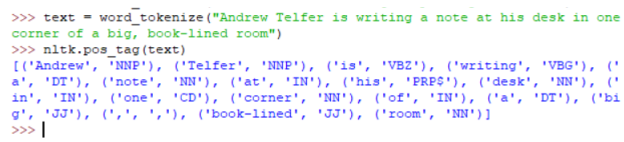

Figure 19: My screen capture of the POS-tagger processing an illustrative utterance from my corpus (i.e. The Ninth Gate Corpus).

There are other, perhaps more attractive, possibilities, apart from the ones described, however, I feel this may suffice to illustrate the point under consideration. In the section that follows, I shall briefly mention certain potential benefits of utilising this free software tool from the point of view of a computational linguist.

Generally speaking, NLP with Python might be regarded as a promising field. It is in this sense that free software tools and libraries enabling such processing are most welcome as precious ingredients of any linguistically-motivated analysis. NLTK represents one such undertaking, which offers a multitude of features for linguists and computational linguists, alike. Corpus-based studies cannot avoid grappling with lexical items present in the naturally occurring language, such as English, French, Serbian, Croatian, etc. Not surprisingly, parsing, tokenisation and part-of-speech tagging have become highly explored possibilities of a computationally-oriented analysis within the current computational linguistic research.

I have shown some illustrative examples performed in the NLTK environment. One may notice potentially useful features, but also the lack of some clear-cut features for some language items, such as orthographic symbols and signs. Even though the part-of-speech tagging is sufficiently felicitous for the majority of computationally-driven analyses, it seems that in some cases, there are certain examples that cannot easily undergo this process. However, these instances seem to be rather rare. This issue poses some challenges, which might be accounted for by the still deeply rooted traditional parts of speech that are treated in terms of necessary and sufficient conditions salient for a lexical item to be included in a given class.

Next, the free software tool NLTK has shown that rather simple programming techniques could be combined so as to deal with large quantities of language material in the form of representative corpora. The features of NLTK abound in different parsing and tagging possibilities and may facilitate the automatic extraction of some key lexical items and/or phrases within a given corpus. Tools and techniques that the Python programming language provides for computational linguistics are numerous, and therefore might represent stimulating challenges posed by natural language processing.

Computing with language, if by this we refer to working with texts (i.e. spoken and written discourses) and words, seems to have been made easier by NLTK and its readily available corpora, followed by some additional features permitting the all-comprising analysis of the language material at hand. Furthermore, free software tool NLTK treats texts as lists of lexical items which may undergo analyses required by a computational linguist, and therefore, may offer help to those computational linguists that need a precise analysis.

III. The Comparison of the Selected Free Software Tools

In the following lines I shall briefly compare three free software tools. Let us briefly consider them in turn. The first free software tool in this analysis is Praat, which is highly functional from the point of view of acoustic phonetics and computational linguistics. Although it has certain limitations as to the duration of the spoken corpus chunk, it certainly represents a reliable resource offering diverse options for a plausible acoustic analysis. Praat has all the advantages of a free software tool and can be easily handled by computational linguists both beginners, and advanced researchers.

The second free software tool in this analysis is KH Coder, which can be used for treating text from a computational point of view, providing all sorts of statistical analyses, both qualitative and quantitative. Even though there are some challenges that should be responded to, such as spelling rules that influence the part of speech tagging and certain lexeme delimitation, these are forgivable weak points in such a multi-perspective analysis provided by KH Coder. This free software tool provides collocation patterns, multi-dimensional analyses and various visualisations which can help and complement the computational analysis of (mainly written) discourse.

The third free software tool in this analysis is NLTK, which is a Python-based natural language toolkit. It has a powerful corpus with the possibility of adding the language input data of one own and abounds in powerful features.

The feature shared by Praat and KH Coder is that there are some limitations with regard to input size. However, sometimes this does not affect performance. Also, one should add that there are some challenges in the NLTK working environment. Namely, when certain tasks are carried out, such as the tokenisation of a larger corpus, the task performance may slow down, and the data displayed after the executed command is not so clear whilst the data manipulation is not straightforward for a computational linguist who is not well-aware of all the possibilities of the Python programming language. However, this is not an insurmountable obstacle on the way paved by NLTK, since it provides other more appealing peculiarities.

Taking as a starting point the notion of performance, the following rough comparison of performance relations between Praat, KH Coder and NLTK might then be posited: NLTK and KH Coder share some features and functionalities (both tools have the possibility of generating and displaying concordances, visualisations, etc.), Praat has visualisation possibilities, but sometimes not of high picture quality. KH Coder and NLTK are

mainly intended for the written medium (i.e. written corpus), whilst Praat is devoted to oral media (i.e. spoken corpus). All three software tools share one common feature, and this refers to the possibility of integrating their tables and graphs readily into scientific papers, conference papers, books, and so on.

The strengths of Praat are to be found in the acoustic analysis of individual sounds, in the annotation of these sounds, and in browsing multiple sound and annotation files across the corpus. The strengths of KH Coder are the visualisations (particularly 3D) that can be further analysed, while the strengths of NLTK lie in its simplicity and elegance of data output display (however, this is in less attractive format than in the case of KH Coder). Pre-processing activities of the analysed software tools have been left aside, although they might also be indicators of certain advantages and disadvantages in raw data processing.

All three software tools have satisfactory output, at least for a user, who is a computational linguist, or a general linguist. It should be added that I have not considered the level of user-friendliness and successfulness from the point of view of a computer scientist, or an electrical engineer, for that matter, but solely from a perspective of a computational linguist. Limitations have been explored solely to a certain extent, since the author of the paper has attempted to perform an analysis by the described free software tools in fairly straightforward corpus-related contexts. Despite the described benefits, it has been noticed, however, that whilst working with large corpora some tools slow down (for instance, NLTK, and KH Coder, whilst processing the data and providing the output of the required feature). Yet, overall impression is that these analysed software tools seem to be irreproachable since they are free of charge and can be further modified and upgraded, which is not the case for proprietary software tools offering the ready-made templates and patterns that cannot be further modified according to one's needs. And this last remark is not insignificant in terms of the last parameter of performance.

The last parameter to be discussed is that pertaining to the user. Namely, certain linguistic research directions are still under the influence of the traditional non-Chomskyan linguistics, and therefore, utilise somewhat different terms and notations which may sometimes indirectly influence some aspects of the linguistic analysis. Praat and KH Coder do not require special programming skills and advanced programming knowledge, whilst NLTK requires sometimes even advanced knowledge of Python. Therefore, a user, who is most frequently a computational linguist, ought to know the fundamentals of this programming language. As regards the corpus-based analysis, it should be highlighted that I used my own corpora for the analyses carried out by means of Praat and KH Coder, whilst I used the ready-made and available corpora in the NLTK environment. Perhaps, this might be the reason for omitting some aspects of analysis since I relied on the previously prepared data. In the part that follows, some concluding and final observations are provided.

IV. Concluding Remarks

In the past six decades or so, we witnessed a rapid growth in the study of what is now well-known as Computational Linguistics. Nevertheless, unitary accounts have been scant. The aim of this investigation is to fill the lacuna in the current scholarship on free software tools in computational linguistics, at least from a descriptive point of view.

The first part of the paper provides introductory remarks and focuses on general observations concerning computers and computational linguistics. Additionally, certain theoretical underpinnings have been mentioned (namely, generative, optimality-driven, relevance-theoretic, and minimalist-motivated, among others). The second part presents free software tools in general, and then presents three free software tools in particular, which served as input to my subsequent argumentation and conclusions. This part is broken into subsections, each of which briefly presents the software tool in question, its performances and potential benefits. The third part is a sketchy comparative analysis which summarises the findings in connection with software tools performances intended for a specific user, i.e. a computational linguist. Some features and functionalities have been compared and a concise overview has been provided.

Although this paper is largely descriptive in its orientation, three case studies reflect the underlying assumptions of the theoretical frameworks in which they are to be found. Equally, this descriptive exploration was aimed at contributing to a better understanding of free software tools in the domain of computational linguistics. Burdened with an ill-famed and notorious reputation of having been persecuted by proprietary software tool creators and distributors, free software tools have not only resisted but are actually struggling for their own place in the realm of computational linguistics. This was illustrated by assessing and evaluating certain striking properties of three free software tools: Praat, KH Coder and NLTK. In the subsequent comparative analysis, these tools were juxtaposed and compared. From the point of view of the user, it has been claimed that expert users tend to operate these tools more easily when compared with linguists. Perhaps, the only exception might be a computational linguist with certain knowledge of programming languages. However, it has been assessed that all analysed tools are user-friendly and can be easily integrated into a linguistically-motivated study.

The analysed and described three software tools can generate graphs and tables and other visualisations that can support any undertaking concerned with linguistic analysis. These visualisations can further refine the analysis in terms of better understanding of relations between the tokens of lexical items. The main area of contention revolves around the questions of the speed of performing certain tasks (e.g. tokenisation of larger corpora, among other things). Another appealing

challenge would refer to the semantic component, which sometimes might not be satisfactorily included in the CTA and NLP, but is, according to the literature, an important ingredient in automatic translation, particularly in scientific fields [82]. Rather than posit these and similar challenges, I have considered the performance of three free software tools within a broader picture of its overall functionality and usefulness in the investigation carried out by a linguist. Therefore, some of my performance measurement results might exclude certain elements that are unimportant for the linguistic analysis.

In this rather brief and unpretentious study, I have reexamined the role of free software tools for computational linguistics from a comparative perspective. To this purpose, I have implemented and analysed three free software tools. My own corpora were used for analysis carried out by Praat and KH Coder, whereas I used the already available corpora and my own examples in the NLTK analysis. Consequently, perhaps this latter decision, to use the already existing language data, might have influenced certain results of the comparative analysis. My observations are not definitive, but rather constitute a tentative descriptive account, which can be further broadened by integrating diverse appealing dimensions of computational text analysis. Some future comparative investigation might significantly contribute not only to our understanding of the role of free software tools in computational linguistics in general, but also of the role of performance-measurement perceived similarities and differences. Needless to say, my tentative assumptions merit further elaboration.

Acknowledgements

I would like to express my gratitude to Professor Predrag Pejović (Faculty of Electrical Engineering, University of Belgrade) for his kindness, patience, expertise and wisdom. I am grateful to Professor Pejović for bringing the free software tools to my research attention. My gratitude goes to Professor Nadica Miljković (Faculty of Electrical Engineering, University of Belgrade) for inviting me to participate in the PSSOH Conference project whilst it was still in the making. Professor Miljković has inspired many parts of my research by generously sharing her experience and knowledge. Both Professor Pejović and Professor Miljković kindly smoothed the way of performing my modest computational analyses. Needless to say, the comments of the Anonymous Reviewer are highly appreciated.

References

[1] M. F. Bott, "Computational Linguistics," *New Horizons in Linguistics*, Harmondsworth, Penguin Books Ltd., 1971, pp. 215-228.

[2] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, New York: Prentice Hall, 2000.

[3] M. Levy, *Computer-Assisted Language Learning: Context and Conceptualization*, Oxford: Clarendon Press, 1997.

[4] N. Smith, *The Twitter Machine: Reflections on Language*, Oxford: Basil Blackwell, 1989.

[5] D. Sperber and D. Wilson, *Relevance: Communication and Cognition*, Oxford: Blackwell, 1988.

[6] J. Lyons, *New Horizons in Linguistics*, Harmondsworth: Penguin Books Ltd., 1971.

[7] N. Chomsky, *Aspects of the Theory of Syntax*, Cambridge, Massachusetts: The MIT Press, 1965.

[8] J. Lieber, *Noam Chomsky: A Philosophic Overview*, New York: St. Martin's Press, 1975.

[9] B. L. Liles, *An Introductory Transformational Grammar*, Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1971.

[10] R. Grishman, *Computational Linguistics: An Introduction*, Cambridge: Cambridge University Press, 1986.

[11] D. Wilson, "Relevance and Relevance theory," *MIT Encyclopedia of the Cognitive Sciences*, Cambridge, Massachusetts , The MIT Press, 1999, pp. 719-722.

[12] R. Kager, *Optimality Theory*, Cambridge: Cambridge University Press, 1999.

[13] D. Abercrombie, *Studies in Phonetics and Linguistics*, London: Oxford University Press, 1965.

[14] A. M. Di Sciullo, "Decomposing Compounds," *SKASE Journal of Theoretical Linguistics*, vol. 2-3, pp. 14-33, 2005.

[15] A. M. Di Sciullo, "Asymmetry and the Language Faculty," *Revista Linguistica*, vol. 13, no. 2, pp. 88-107, 2017.

[16] N. Smith and D. Wilson, *Modern Linguistics: The Results of Chomsky's Revolution*, Harmondsworth: Penguin Books Ltd., 1979.

[17] J. Pustejovsky, *The Generative Lexicon*, Cambridge, Massachusetts; London, England: The MIT Press, 1995.

[18] D. A. Reibel and S. A. Schane, *Modern Studies in English: Readings in Transformational Grammar*, Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1969.

[19] R. A. Jacobs and P. S. Rosenbaum, *Readings in English Transformational Grammar*, Waltham, Massachusetts: Ginn and Company, A Xerox Company, 1970.

[20] N. Chomsky, *The Minimalist Program*, Cambridge, Massachusetts; London, England: The MIT Press, 1995.

[21] A. M. Di Sciullo and E. Williams, *On the Definition of Word*, Cambridge, Massachusetts: The MIT Press, 1987.

[22] D. Davidson, "Semantics for Natural Languages," *On Noam Chomsky: Critical Essays*, G. Harman, Ed., New York, Anchor Press, pp. 242-252, 1974.

[23] D. Burton, "Analysing Spoken Discourse," *Studies in Discourse Analysis*, M. Coulthard and M. Montgomery, Eds., London, Routledge and Kegan Paul Ltd., pp. 61-81, 1981.

[24] M. Coulthard, M. Montgomery and D. Brazil, "Developing a Description of Spoken Discourse," *Studies in Discourse Analysis*, M. Coulthard and M. Montgomery, Eds., London, Routledge and Kegan Paul Ltd., pp. 1-50, 1981,.

[25] D. Sperber, F. Clément, C. Heintz, O. Mascaro, H. Mercier, G. Origgi and D. Wilson, "Epistemic Vigilance," *Mind and Language*, vol. 25, no. 4, pp. 359–393, 2010.

[26] D. Mazzarella, *Inferential Pragmatics and Epistemic Vigilance*, London: University College London, 2015.

[27] J. Feller, B. Fitzgerald, S. Hissam and K. R. Lakhani, "Introduction," *Perspectives on Free and Open Source Software*, Cambridge, Massachusetts, The MIT Press, 2005, pp. xxvii-xxxi.

[28] H. T. Le and J. Brook, "Using Praat to Teach Intonation to ESL Students," *Hawaii Pacific University TESOL Working Paper* Series 9, vol. 9, no. 1-2, pp. 2-15, 2011.

[29] H. Buschmeier and M. Włodarczak, "TextGridTools: A TextGrid Processing and Analysis Toolkit for Python," *Proceedings der 27. Konferenz zur Elektronischen Sprachsignalverarbeitung*, Bielefeld, 2013.

[30] P. Boersma and D. Weenink, *Praat: Doing Phonetics by Computer*, 6 June 2020. [Online]. Available: https://www.fon.hum.uva.nl/praat/. [Accessed 6 July 2020].

[31] M. S. Suri, D. Setia and A. Jain, "PRAAT Implementation For Prosody Conversion," *Proceedings of the 4th National Conference; INDIACom-2010; Computing For Nation Development*, New Delhi, 2010.

[32] D. Loakes, "From IPA to Praat and Beyond," *The Oxford Handbook of the History of Linguistics*, Oxford, Oxford University Press, pp. 123-140, 2013.

[33] M. Magdin, T. Sulka, J. Tomanová and M. Vozár, "Voice Analysis Using PRAAT Software and Classification of User Emotional State," *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 5, no. 6, pp. 33-42, 2019.

[34] P. Boersma, "Optimality-Theoretic Learning in the Praat Program," *Institute of Phonetic Sciences Proceedings*, Amsterdam, 1999.

[35] J. Setter and J. Jenkins, "State-of-the-Art Review Article: Pronunciation," *Language Teaching*, vol. 38, no. 01, pp. 1-17, 2005.

[36] I. Wilson, "Using Praat and Moodle for Teaching Segmental and Suprasegmental Pronunciation," *TESOL Quarterly*, vol. 3, no. 1, pp. 33-43, 2005.

[37] B. Gorjian, A. Hayati and P. Pourkhoni, "Using Praat Software In Teaching Prosodic Features To EFL," *Procedia - Social and Behavioral Sciences*, Izmir, 2013.

[38] P. Boersma and V. v. Heuven, "Speak and unSpeak with PRAAT," *Glot International*, vol. 5, no. 9-10, pp. 341-347, 2001.

[39] J.-P. Goldman and A. C. Simon, "ProsoBox: a Praat Plugin for Analysing Prosody," *Acts of the 10th Speech Prosody Symposium*, Tokyo, 2020.

[40] W. Styler, "Using Praat for Linguistic Research," 25 December 2017. [Online]. Available: http://wstyler.ucsd.edu/praat/UsingPraatforLinguisticResearchLatest.pdf. [Accessed 6 June 2020].

[41] N. Chomsky and M. Halle, *The Sound Pattern of English*. *The First MIT Press Paperback Edition. Third Printing*, Cambridge, Massachusetts: The MIT Press, 1995.

[42] L. Bauer, "When is a Sequence of Two Nouns a Compound in English?," *English Language and Linguistics*, vol. 2, p. 65–86, 1998.

[43] L. Bauer, "Adjectives, Compounds and Words," *Nordic Journal of English Studies*, vol. 3.1, Special Issue: Worlds of Words: A Tribute to Arne Zettersten, p. 7–22, 2004.

[44] I. Plag, "The Variability of Compound Stress in English: Structural, Semantic, and Analogical Factors," *English Language and Linguistics*, vol. 10, no. 1, p. 143–172, 2006.

[45] L. Bauer, *Compounds and Compounding*, Cambridge: Cambridge University Press, 2017.

[46] M. Johnston and F. Busa, "Qualia Structure and the Compositional Interpretation of Compounds," *Proceedings of the ACL SIGLEX Workshop on Breadth and Depth of Semantic Lexicons*, Santa Cruz, California, 1996.

[47] I. Plag, G. Kunter and S. Lappe, "Testing Hypotheses about Compound Stress Assignment in English: A Corpus-Based Investigation," *Corpus Linguistics and Linguistic Theory*, vol. 3, no. 2, pp. 199-232, 2007.

[48] I. Plag, "Compound Stress Assignment by Analogy: The Constituent Family Bias," *Zeitschrift für Sprachwissenschaft*, vol. 29, no. 2, pp. 243-282, 2010.

[49] M. J. Bell and I. Plag, "Informativeness is a Determinant of Compound Stress in English," *Journal of Linguistics*, vol. 48, no. 3, pp. 485-520, 2012.

[50] M. Johnston, B. Boguraev and J. Pustejovsky, "The Acquisition and Interpretation of Complex Nominals," *Working Notes of AAAI Spring Symposium on the Representation and Acquisition of Lexical Knowledge*, Stanford, California, 1995.

[51] L. Polanyi, *Telling the American Story: A Structural and Cultural Analysis of Conversational Storytelling*, Cambridge, Massachusetts: The MIT Press, 1989.

[52] K. Higuchi, "Introduction to KH Coder," 6 June 2020. [Online]. Available: https://khcoder.net/en/. [Accessed 6 7 2020].

[53] S. A. García, N. O. Martínez, T. C. Carabel and I. F. Suárez, "Occupational Accidents and Their Prevention in the Spanish Digital Press," *Revista Latina de Comunicación Social*, vol. 72, pp. 1608-1625, 2017.

[54] K. Takamatsu, Y. Kozaki, A. Kishida, K. Bannaka, K. Mitsunari and Y. Nakata, "Analyzing Students' Course Evaluations Using Text Mining: Visualization of Open-Ended Responses in Co-Occurrence Network," *PEOPLE: International Journal of Social Sciences*, vol. 4, no. 3, pp. 142-153, 2018.

[55] N. Wang, "Potential for Teaching Materials for Advanced Reading Comprehension with the Use of Text Mining: A Report on the Practice of Using KH-Coder," *Senshu University Institute of Humanities Monthly Bulletin*, vol. 304, no. 1, pp. 19-29, 2020.

[56] R. Shineha and M. Tanaka, "Deprivation of Media Attention by Fukushima Daiichi Nuclear Accident: Comparison Between National and Local Newspapers," *Resilience: A New Paradigm of Nuclear Safety - From Accident Mitigation to Resilient Society Facing Extreme Situations*, J. Ahn, F. Guarnieri and K. Furuta, Eds., Cham, Springer International Publishing, pp. 111-125, 2017,.

[57] O. Ylijoki and J. Porras, "Conceptualizing Big Data: Analysis of Case Studies," *Intelligent Systems in Accounting, Finance and Management*, vol. 23, no. 4, pp. 295-310, 2016.

[58] P. Nattuthurai and A. Aryal, "Content Analysis of Dark Net Academic Journals from 2010-2017 Using KH Coder," *ACET Journal of Computer Education and Research*, vol. 12, no. 1, pp. 25-35, 2018.

[59] K. Benoit and A. Herzog, "Text Analysis: Estimating Policy Preferences from Written and Spoken Words," *Analytics, Policy and Governance*, J. Bachner, B. Ginsberg and K. Hill, Eds., New Haven, Connecticut, Yale University Press, pp. 137-159, 2017.

[60] S. Hori, "An Exploratory Analysis Of The Text Mining Of News Articles About "water And Society"," *WIT Transactions on The Built Environment*, vol. 168, pp. 501-508, 2015.

[61] S. Palmer and M. Campbell, " Text Analytics Visualisation of Course Experience Questionnaire Student Comment Data in Science and Technology," *AAEE 2015: Proceedings of the Australasian Association for Engineering Education 2015 Annual Conference*, Geelong, Victoria, 2015.

[62] M. D. Đurić, "Some Aspects of the Discourse Pertaining to Digital Musems of Digital Art," *MELISSA - Museums, Ethics, Library and Information Science, Studies, Archives*, vol. 16, no. 1, pp. 125-146, 2017.

[63] D. Biber, *Variation across Speech and Writing*, Cambdridge: Cambridge University Press, 1995.

[64] C. Krstev, *Processing of Serbian: Automata, Texts and Electronic Dictionaries*, Belgrade: Faculty of Philology, University of Belgrade, 2008.

[65] R. Carston, "Language and Cognition," *Linguistics: The Cambridge Survey, 3: Language: Psychological and Biological Aspects*, Cambridge, Cambridge University Press, pp. 38-68, 1989.

[66] D. Vitas, S. Koeva, C. Krstev and I. Obradović, "Tour du monde through the dictionaries," *Actes du 27eme Colloque International sur le Lexique et la Grammaire*, L'Aquila, 2008.

[67] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," *Proceedings of the ACL-02 Workshop onEffective tools and methodologies for teaching natural language processing and computational linguist*, Philadelphia, Pennsylvania, 2002.

[68] S.Vijayarani and R.Janani, "Text Minijng: Open Source Tokenization Tools - An Analysis," *Advanced Computational Intelligence: An International Journal (ACII)*, vol. 3, no. 1, pp. 37-47, 2016.

[69] S. Bird, E. Klein, E. Loper and J. Baldridge, "Multidisciplinary Instruction with the Natural Language Toolkit," *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics (TeachCL-08)*, Columbus, Ohio, 2008.

[70] S. Bird, E. Klein and E. Loper, *Natural Language Processing with Python*, Sebastopol, California & Cambridge: O'Reilly Media, Inc., 2009.

[71] J. Perkins, *Python Text Processing with NLTK 2.0 Cookbook*, Birmingham: Packt Publishing Ltd., 2010.

[72] S. Bromberger, "Types and Tokens in Linguistics," *Reflections on Chomsky*, A. George, Ed., Oxford, Basil Blackwell, pp. 58-89. 1990.

[73] S. Bromberger, *On What We Know We Don't Know: Explanation, Theory, Linguistics, and How Questions Shape Them*, Chicago; London; Stanford: The University of Chicago Press; Center for Study of Language and Information, 1992.

[74] P. H. Matthews, *The Concise Oxford Dictionary of Linguistics*, Oxford: Oxford University Press, 2005.

[75] H. G. Widdowson, *Linguistics*, Oxford: Oxford University Press, 1996.

[76] S. Bird, E. Klein and E. Loper, "Language Processing and Python," 2009. [Online]. Available: http://www.nltk.org/book_1ed/ch01.html. [Accessed 2 August 2020].

[77] D. Crystal, *A First Dictionary of Linguistics and Phonetics*, Second Impression, London: Andre Deutsch, 1983.
[78] F. Palmer, *Grammar*. Second Edition, Harmondsworth: Penguin Books Ltd.,1986.
[79] J. Aitchison, *Cassell's Dictionary of English Grammar*, London: Cassell & Co., 2001.
[80] R. L. Trask, *The Penguin Dictionary of English Grammar*, Harmondsworth: Penguin Group Ltd., 2000.
[81] S. Pinker, *The Language Instinct: How the Mind Creates Language*, New York: HarperPerennial, 1995.
[82] S. Thomas, *Computers: Their History, Present Applications, and Future*, New York: Holt, Rinehart and Winston, Inc., 1965.