The Thesis committee for Juan Daniel Pinto
Certifies that this is the approved version of the following thesis:

Creating a Conversational Hebrew Vocabulary List

APPROVED BY

SUPERVISING COMMITTEE:

Esther Raizen, Supervisor

Elaine Horwitz, Co-Supervisor

Creating a Conversational Hebrew Vocabulary List

by

Juan D. Pinto

Thesis

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Arts in Hebrew linguistics

The University of Texas at Austin
May 2018

Dedication

Dedicated to

Acknowledgements

Interdum et malesuada fames ac ante ipsum primis in faucibus. Aliquam congue fermentum ante, semper porta nisl consectetur ut. Duis ornare sit amet dui ac faucibus. Phasellus ullamcorper leo vitae arcu ultricies cursus. Duis tristique lacus eget metus bibendum, at dapibus ante malesuada. In dictum nulla nec porta varius. Fusce et elit eget sapien fringilla maximus in sit amet dui.

Mauris eget blandit nisi, faucibus imperdiet odio. Suspendisse blandit dolor sed tellus venenatis, venenatis fringilla turpis pretium. Donec pharetra arcu vitae euismod tincidunt. Morbi ut turpis volutpat, ultrices felis non, finibus justo. Proin convallis accumsan sem ac vulputate. Sed rhoncus ipsum eu urna placerat, sed rhoncus erat facilisis. Praesent vitae vestibulum dui. Proin interdum tellus ac velit varius, sed finibus turpis placerat.

Creating a Conversational Hebrew Vocabulary List

by

Juan Daniel Pinto
The University of Texas at Austin, 2018
SUPERVISORS: Esther Raizen, Elaine Horwitz

Indent and begin abstract here. It should be a concise statement of the nature and content of the ETD. The text must be either double-spaced or 1.5spaced. Abstracts should be limited to 350 words.

Table of Contents

List of tables

List of figures

1 Introduction

This thesis explains the theory behind the creation of the Conversational Hebrew Vocabulary List (hereafter CHVL)—a Modern Hebrew high-frequency word list—along with implications from the project. The list itself is included as an appendix, and can also be downloaded free of charge in a variety of formats. While evaluating past methods for the creation of such lists, it became clear that a large gap in the literature exists when it comes to less commonly taught languages (LCTLs). Because the overwhelming majority of the previous research in this area has been focused on English alone, some important nuances are yet to be addressed. More often than not, the few non-English word lists that do exist, along with much of the research in vocabulary acquisition, have taken at face value some of the findings of this Anglo-centric research, often without questioning whether the same methodologies and conclusions apply to different languages.

The present paper is, therefore, an effort to help educators interested in creating and/or using word lists for their own classrooms, for wider dissemination, or simply for general research purposes. In doing so, it will simultaneously provide an overview of some of the key decisions that must be taken into account for such a project, along with key studies on the topic.

The various uses of word frequency lists can be roughly classified into research applications and practical applications. Examples of research applications include traditional linguistic studies that look for common morphological patterns, corpus-linguistic studies seeking to understand language through "real world" texts, and psycholinguistic studies that explore connections between a speaker's mental lexicon and word frequency. Practical applications of word lists include curriculum and text-book planning for language teachers, vocabulary selection for graded readers and dictionaries, and even independent language study. Of course, the line between research and practical applications can be rather fuzzy. Some of the most important studies lie between these two groups, and attempt to answer questions such as: How can vocabulary knowledge be appropriately tested and measured? What is the role of extensive reading (as opposed to intensive reading) in incidental vocabulary acquisition? What level of vocabulary do learners need in order to read extensively for pleasure? What level of vocabulary do learners need in order to succeed in an academic setting? What role does specialized vocabulary play in reaching understanding? These questions and their answers rely heavily on the creation and use of trustworthy word frequency lists. Yet due to the resources and effort required to create these lists, they are rarely found in languages other than English.

The theoretical foundation for the creation and use of word frequency lists rests on the observation, made popular by the linguist George Kingsley Zipf in the 1930s and 40s, that if one were to create a frequency list of words in a large enough text, the first word would occur roughly twice as often as the second word, three times as often as the third word, and so on (Zipf 1935, 1949). This exponential distribution is significant because it means that a small number of words make up the bulk of a text, whereas the majority of the words occur very few times. Paul Nation, one of the most influential scholars in the field of vocabulary acquisition, has pointed out that Zipf's Law—as it is has come to be known—can serve as motivation to language learners and teachers, since learning the most common vocabulary in a language covers so much of the communication that naturally occurs (2001).

The primary research question guiding this project is this: *What are the most-frequently used words in conversational Modern Hebrew?* The resulting study also addresses the following secondary research questions, which were necessary to address in order to answer the aforementioned question: *What effect does a corpus of unvocalized texts have on the identification of word families in the computerized creation of a vocabulary frequency list? What factors affect the way that boundaries are demarcated for various levels of word families in Modern Hebrew?* And finally: *What implications might these findings have for word list creation and use as it pertains to other less commonly taught languages?*

The literature review will serve as a basis for many of the important decisions taken during the creation of the CHVL. These decisions—surrounding both corpus and list creation—along with their reasoning, will be explained further in an analysis of the literature. For the sake of clarity, these decisions are listed here at the outset. They are as follows:

Corpus design - *Size:* - *Text types:* The corpus consists of a single text type: conversation. This is to best fit with the list's intended audience. In order to accomplish this, movie and television subtitles compose the core of the corpus. **List design:** - *Use:* The primary intended audience for the CHVL is composed of beginning-to-low-intermediate learners of Hebrew as a foreign language. It is designed for both receptive and productive language use. - *Word family levels:* The word family level that is best suited for the CHVL's intended audience is the lemma. - *Criteria:* The CHVL was created using exclusively objective criteria, meaning that it is the product of calculations, and it was not manually tweaked in any way. The empirical criteria used were frequency and range.

Following the review of literature and explanation of theory, the process of the CHVL's creation will be explained in detail, along with findings from the project. Possible implications for other less commonly taught languages will then be discussed. Finally, the full CHVL will be provided as an appendix.

2 Review of the literature

2.1 Corpus Design

For a word list to accurately reflect the use of a language in its broadest context, the corpus from which it is extracted needs to be representative of that context. Since it is impossible to analyze all of the communications that take place in a particular language (not even taking into account the fact that language itself is an ever-expanding, ever-changing, *open* corpus), researchers must make do with an approximation of the whole: a bounded corpus of language.

Though the focus of this literature review is the creation of word frequency lists, the truth is that relatively few corpora have been created for this specific purpose. Most corpora have aimed at being general collections that cover the language (usually English) as a whole in an attempt to serve different theoretical and applied uses. Yet despite this broad range of purposes, the creation of corpora has historically revolved around two big questions: (1) how large should the corpus be, and (2) what kinds of texts should it include? I will here address these two points separately, with the recurring emphasis remaining on corpus use for word list creation.

2.1.1 Corpus Size

Conventional wisdom in corpus creation states that more is better. If a word list is to accurately reflect the frequencies of words in the language as a whole, then a corpus must contain enough text to approximate the overall use of discourse. This line of thinking is equivalent to the maxim in quantitative research that a sample should be as representative of the target population as possible. And in order to maximize the statistical probability of this representativeness, the sample must be of an appropriate size for the study. True, larger sample sizes often increase this probability, but they also tend to be more resource-intensive for the researcher. The same is true of corpus size. When creating a vocabulary list, then, what is an “appropriate” corpus size?

Corpora composed of millions of tokens are easy to access today. This is especially true of corpora of written material—corpora of spoken language are still comparatively small. Advances in computing power have made it possible to analyze these mega-corpora, something that would have been far too labor-intensive in the not-so-distant past. It is finally becoming plausible for more researchers without access to extensive resources to use these mega-corpora for the purpose of word list creation.

The first project to create a one-million-token corpus was Kučera and Francis' effort at Brown University to compile a corpus of American English texts printed in 1961. They strived to create a corpus with equal amounts of texts from different sources by randomly selecting 500 passages of 2,000 words each from different published materials found at the Brown University Library and the Providence Athenaeum. This mixed design would be used as a model by many of the corpora created during the next few decades: . These began to be compiled at increasingly faster rates. Many of these corpora were created—in part—to serve as parallel corpora of different varieties of English.

What began in 1980 as a collaboration between Collins Publishing and a group of researchers—the *Collins Birmingham University International Language Database* (COBUILD)—became a 7-million-token corpus by 1982. It continued expanding until it was joined to *The Bank of English* corpus in the 1990s, which reached 320 million words in 1997. In 2004, as part of the Collins World Web, it reached 2.5 billion words (HarperCollins Publishers, 2004a, 2004b). Now, with the use of web-crawling applications that scour the internet and collect text at unprecedented speed, there exist English corpora 11 billion tokens (*enTenTen12*) and even 19 billion tokens (*enTenTen13*).

Clearly, then, the sky's the limit when it comes to ever-growing corpora of language. But when it comes to word list creation, is there a corpus size that can be considered sufficient?

Studies have approached this specific problem of corpus size for word list creation by creating multiple frequency lists—each from a different size of corpus—and then comparing the efficacy of these lists themselves. But what makes a word list effective? Different researchers have tackled this question differently.

One way to judge the effectiveness of a word list is to find how closely it correlates with reaction times in a lexical decision task—a widely-used procedure in psychological and psycholinguistic research. In a lexical decision task, participants are presented with a series of words and non-words, one after the other, and they are asked to judge which is which as quickly as possible. The reaction times are then analyzed for each word. The basic idea behind this experiment is that the average time it takes participants to react to a word reflects something about the way the word is accessed in participants' mental lexicons. Given enough data, it is possible to make certain inferences about the arrangement of this internal lexicon, which has led to various psycholinguistic theories over the years. But what does this have to do with words on a frequency list? There is well-attested evidence to suggest that there exists an inverse correlation between word frequency and reaction time on a lexical decision task (Whitney, 1998; Balota and Chumbley, 1984). In other words, more common words are accessed and recognized more quickly than less common words. Therefore—the thinking goes—an effective word frequency list should correspond to and reflect this reality.

This was precisely the approach taken by Brysbaert & New (2009), who compared respond times collected as part of the massive Elexicon Project (Balota, et al., 2007) to words on a series of frequency lists made from increasingly larger corpora. The corpora used were all subcorpora extracted from the British National Corpus (BNC). With each subsequent increase in token count, the word list correlated more and more closely with the response times from lexical decision tasks. This observation validates the line of thinking described at the beginning of this section regarding the need for large corpora. Brysbaert and New hoped to find an "ideal" corpus size after which the increase in effectiveness would no longer be significant enough to justify the additional cost of resources. After conducting several regression analyses on the two sets of data, they found that the variance in the response times that could be accounted for by corpus size reached a plateau at about 16 million words. In other words, for corpora with less than 16 million words, the size of the corpus had a significant effect on the correlation between word frequencies and average response times for those words on lexical decision tasks. For corpora with more than 16 million words, the effect of increasing corpus size became considerably more subtle. In the end, they concluded that in order to construct an effective word list for *high-frequency* words, a corpus of about 1 million tokens is needed. However, in order to reach the same effectiveness for *low-frequency* words, a corpus size of at least 16 million words is preferable.

A different, more straightforward methodology is to directly compare word lists made from corpora of different sizes. Rather than judging the "effectiveness" of a list, this approach measures similarities shared between different lists. Hypothetically, doing this at increasing corpus sizes should allow one to find a size after which the variance between lists only minimally decreases. As with the previous approach, the goal here is to find a point at which the benefits of increasing size no longer outweigh the additional needed resources.

Essentially, then, all corpora of sufficient size should result in nearly the same word frequency list—a theory based on a strict interpretation of Zipf's law applied to all natural language. If the appropriate criteria can be found—Sorell (2013) suggests—then this would, at last, provide a solution to Nation's (2001, 2013) observation that, problematically, word lists tend to disagree rather drastically on both the words included and their respective ranking.

Inspired by the computational linguistic measure of *rank distance* (Popescu and Dinu, 2008)—a method for comparing stylistic differences between texts—Sorell (2013) developed a variant of this methodology. First, he used different corpora of the same size to create multiple word lists, one for each corpus, ranked entirely by frequency. He then identified the percentage of words that are *not* shared between each set of two lists. Finally, he averaged these percentages to find the level of variability created at that specific corpus size. The levels of variability he found were remarkably close to each other—despite using a wide variety of entirely different corpora (with no overlap on texts within each one). He then increased the size of each corpus and repeated the process.

In order to calculate this level of variability, Sorell used a modified version of a complex formula that he borrowed from the natural sciences, and called his resulting calculation the *Dice distance*. Though this Sørensen–Dice coefficient that he altered (also known by other names) is widely used in botany and other fields to measure similarity in areas and samples of different sizes, the frequency lists measured by Sorell were all purposefully of the same size. What this means is that—apparently without realizing it—his *Dice distance* was ultimately just a simple percentage: *number of different words between frequency lists / size of frequency list (total words).*

Regardless of the round-about way he used to calculate it, his resulting measure for each corpus size—the level of variability—can be accurately described as the average proportion of difference for word lists at that particular corpus size.

Sorell found that a stable list (about 2% variation) of the most frequent 1,000 words, or a reasonably stable list (less than 5% variation) of the most frequent 3,000, words can be created using a corpus of 50 million tokens. In other words, 1,000-type word lists created from different 50-million-token corpora will likely only differ by 20 words. At the 3,000-type level using the same sizes of corpora, the lists will likely vary by less than 150 words. This is a remarkable level of similarity. Expanding the list to 9,000 types will still only have about 4–7% variation, or 360–630 words. Even corpora of 20 million tokens can be considered sufficient in many cases, since they will result in 3,000-type word list with roughly 5% variation, and 9,000-type word list with less than 10% variation.

Taking a similar approach, though with significant variations, Brezina and Gablasova (2015) compared four corpora of various sizes: The Lancaster-Oslo-Bergen Corpus (LOB), The BE06 Corpus of British English (BE06), The British National Corpus (BNC), and EnTenTen16. These corpora had respective token sizes of 1 million, 1 million, 100 million, and 12 billion. The word list created from each corpus was, in this case, a combination of frequency and dispersion—a measure that will be discussed in more detail later in this paper. The resulting word lists were then compared, and the percentage of shared vocabulary words calculated. Additionally, the researchers also calculated the correlation between the ranking for each word that was shared between word lists. Contrary to Sorell, Brezina and Gablasova considered this final comparison an important part of understanding the effect of corpus size.

The aim of this study was not to find a corpus size after which the difference was negligible, but rather to find if there was a significant difference between word lists made from corpora of different sizes. The study found a 78%–84% overlap between each of the 3,000–lemma word lists. 71% of the words were shared among all four of the lists. Based on this number, Brezina and Gablasova concluded that regardless of corpus size—at least for anything larger than one million tokens—“similar results” are obtained.

This conclusion differs significantly from Sorell’s, who concluded that a corpus of at least 20 million tokens (though 50 million is preferable) is needed for a stable word list with low variability. These disagreements are primarily the result of a difference in what should be considered "stable." At 71% vocabulary overlap—which is sufficient for Brezina and Gablasova—870 words were only found in one of the four lists. This is drastically higher than Sorell's threshold, which at the 3,000-word level varies in roughly 150 words. Note that Nation and Hwang (1995) found a level of overlap similar to Brezina and Gablasova when comparing the GSL, the LOB, and the Brown corpora—a percentage of overlap that they deemed to be not particularly high. As Nation later put it, "Brezina and Gablasova are a bit too tolerant in accepting that 71% or even 78%-84% overlap is good enough. If roughly one out of every four or five words is different from one list to another, that is a lot of difference" (2016, p. 100).

Another difference to mention between these two studies is the unit of counting used. Sorell made lists based on *types*, whereas Brezina and Gablasova preferred the use of *lemmas*. I will explain this important distinction in a later section of this review ("Identifying Words"). For now, it is sufficient to say that the effect of these different measures in comparing word lists created from corpora of different sizes has (to my knowledge) not been studies. This is one area that could benefit from further research.

Lastly, the corpora used by Brezina and Gablasova were all-inclusive: each built on its own philosophy on the way that different types of texts should be balanced in a corpus, but all seeking to be representative of English as a whole. This is also true of the corpora used by Brysbaert and New in their study using response times from a lexical decision task. Contrast this with Sorell's word lists, which were systematically created from corpora that consisted of only one specific text type. Surely, this is a factor to consider in corpus design.

Therefore, having a sufficiently large corpus is important, as demonstrated in this section. But is it enough? How much do the types of texts included in a corpus factor into its effectiveness for word list creation?

2.1.2 Text Types

There's been a lot of debate about the "best" way to balance a corpus' text types. This is a major aspect of corpus design, and one worth delving into. At the end of the day, much of it comes down to the purpose of the corpus. When used for the creation of word lists, one must also consider the intended purpose of the word list itself. Is it for general use or for one of many possible specialized uses? More on this in the next section.

In order to design a corpus with different amounts of text types (i.e. narrative, conversational, academic), clear definitions for these text types are necessary. But is there a better way than the use of subjective genres to classify texts?

Or is there a better methodology than simply mixing a bunch of different texts together, with the hope that the resulting word list covers the language as a whole? This is the most common way of creating frequency lists, but it tends to result in a mix of words that have little relevance to any one purpose. Esoteric, academic words in a beginners' vocabulary list? Science fiction terms in a vocabulary list for business managers? It's obvious that a list is only as good as the corpus from which it's made, which is why a clear delineation of different text types and their qualities is critical.

When speaking of corpus balance, I refer to the proportion of different text types that make up a corpus. Published corpora have taken different approaches in this regard, and published word lists have made use of a variety of strategies for balancing the corpora from which they are made. Coxhead's *Academic Word List* (2000) was created from a carefully-designed corpus that used equally-sized sub-corpora of texts from different disciplines. This suited the purpose of her word list well, since it was intended to serve students from a variety of disciplines.

The importance of identifying a taxonomy of text types based on objective criteria: are there distinguishable linguistic differences between an informal correspondence and a narrative work of fiction? What about between a romance and a fantasy novel?

Biber's early work (1988) conducted an analysis of a wide variety of texts using large corpora to tag syntactic markers and other linguistic attributes that could potentially be used to define different types of texts. In this study, he found a series of five categories (each consisting of two opposite ends of a spectrum) in which texts varied: involved vs. informational, narrative, situated vs. elaborated, persuasive, and abstract. He then conducted a very large study, which he published as a book, (1995) that found eight distinct, recurring patterns of different combinations of these categories. These groupings serve as a linguistically-based taxonomy that divides texts along objective lines, rather than subjective, culturally-defined genres.

Similar but independent studies were conducted for Somali, Korean, Nukulaelae Tuvuluan, Taiwanese, and Spanish (Biber, 1995; Jang, 1998). For each language, a unique set of text types were identified. However, the texts were found to align along similar distinguishing linguistic dimensions as the English texts.

Sorell (2013) sought to simplify Biber's eight text types into categories suitable for corpora study. He did this by noticing the closely similar ways that some of the text types lined up along Biber's five linguistic categories, also incorporating some extra-linguistic features, such as shared contexts (e.g. predominantly spoken types). He also dropped Biber's two smallest text types, deeming them impractical for corpus study and difficult to isolate. In doing this, he came up with four simplified text types: interactive (conversation), general reported exposition (general writing), imaginative narrative (narrative writing), and academic. Regarding this last type, Biber's study found a sosignificant difference between academic writing in the natural sciences ("scientific exposition") and the humanities ("learned exposition")—he found that natural science uses more concrete language, whereas the humanities tend to use more abstract language. However, Sorell sought to unify these for the sake of simplicity, simply leaving their distinction to "a future study" (p. 68). Sorell acknowledged that his wasn't the first attempt at simplification of Biber's text types, a surprisingly similar effort having been made in the *Longman Grammar of Spoken and Written English* (Biber, Johansson, Leech, Conrad, & Finegan, 1999: 16) and the *Longman Student Grammar of Spoken and Written English* (Biber, Conrad, & Leech, 2002: 23).

Sorell found that each of his four simplified text types yielded a vocabulary frequency list that was as unique as the linguistic criteria that Biber had used. He also measured how different they were from each other, and found all four to be equidistant from the next in this order: conversation, narrative, general writing, and academic writing (See section on corpus size for an explanation of this measurement). Sorell, therefore, claims that his own study of vocabulary frequency using his simplified text types as a base has “validated Biber’s studies by adding a vocabulary dimension to the description of each of the key text types” (201).

Despite the importance of spoken language—or the conversation text type—for language learners and linguistic studies, the number of conversation corpora that exist, as well as their size, is very limited. This is clearly because of the difficulty of gathering large amounts of spoken data that then needs to be transcribed by hand in order to be analyzed. It is true that speech recognition software has come a long way in recent years, but its rate of error remains too high for research purposes. It has been estimated that it takes 40 hours to professionally transcribe one hour of audio recording, making the task too costly. For this reason, some researchers have begun looking at alternative sources for a conversation corpus, including the internet and movie subtitles.

New, et al. (2007) created a 50-million-token corpus of French subtitles. They divided this into four subcorpora, one for each of the type of media from which the subtitles were extracted: French films, English movies, English television series, and non-English-language European films. The reason for using French subtitles from English media is the sheer dominance of English in the film industry. In order to counter-balance the much larger sizes of the two subcorpora extracted from English media, the researchers measured word frequencies for each subcorpora separately, then averaged them to arrive at the final frequency used for their ranked word list.

In order to test the validity of their new approach, New, et al. used two different methods. First, they compared their subtitle word list with word lists created from more traditional corpora. Second, they used lexical decision times—similar to Brysbaert and New (2009) above—to test the rankings of words on their list.

The first test found a .73 correlation with a classical French spoken corpus, the “Corpus de Référence du Français Parlé” (CRFP; Equipe DELIC, 2004). However, when looking at the specific words and semantic categories that differ the most, it’s clear that most major differences are caused by the monologue-nature of the CRFP. This corpus was created from a large number of interviews (each asking the same questions to the interviewee), whereas movie subtitles tend to be composed primarily of people interacting in conversations. This results in more colloquial expressions having higher frequencies in the subtitle corpus. The nature of movies themselves also played a role, resulting in an overrepresentation of words related to action movies and police matters—words like *tuer* [to kill], *prison* [jail], and *armes* [weapons] (p. 665).

For the second test of the subtitle word list, the researchers used the lexical decision times from two previous experiments. They found that the subtitle list's ability to predict lexical decision times was at least equally as accurate as the CRFP frequencies or those from a traditional corpus of written French. In many cases, it actually fared much better, surprising even the researchers themselves. However, this latter test was based on the rather small sample sizes of the two previous experiments (234 and 240 words), limiting the reliability of this test.

Picking up on these findings, and expanding the lexical decision task to a much larger sample size, Brysbaert and New (2009) compiled a corpus of English subtitles (SUBTLEX$_{\mathrm{US}}$) and evaluated it as part of their study. This corpus is composed of subtitles from a wide variety of American films since 1900, though a majority are from 1990, as well as a large number of American television series. They found that the subtitle frequencies were especially good at predicting the lexical decision times of short words, often surpassing the accuracy of rankings based on the many written corpora they tested. It had more difficulty explaining the response times of longer words, which are more rarely found in film than in literature. Overall, their own conclusion confirmed that of the New, et al. (2007) study, that word frequencies derived from subtitle corpora seem to have a clear advantage over other types of corpora.

Though these two studies arrive at the same conclusion regarding the use of subtitles, more research is needed in this area. If, indeed, subtitles can be considered as appropriate sources for corpora of the conversation text type, their availability will open many possibilities previously made nearly impossible by the difficulty of the collection medium.

2.2 List Design

Perhaps even more complex than appropriately designing the corpus from which to extract vocabulary for a word list, researchers have found a wide range of variables that play a role in the design of the list itself. Questions addressed in the literature deal with the difference between a general service list and a specialized list, differences in the way that a "word" is defined and measured, different ranking criteria used, and the influence of subjective criteria on list creation, among other issues.

2.2.1 General Use vs. Specialized Use

Nation (2016) emphasized the importance of identifying the purpose of a word list before beginning the creation process. He believes that the main purpose of most general-use lists is to select vocabulary that language learners should learn during their first years of study. Though this may be the stated goal of some general-use lists, it is clear that they in fact serve a wide variety of purposes. He rightfully suggests, however, that the goal of serving language learners is far too broad to be very helpful. Language learners come to the task at different ages, with different language needs, and with different reasons for learning the language. A word list that is useful for adult learners intent on attending university will likely not be helpful for young leaners whose language focuses on animals, colors, and other age-appropriate material. And yet general-use lists are far more common than specialized-use lists. This is largely due to attempt at finding the language's core vocabulary.

The majority of word lists in use attempt to describe the vocabulary of the language as a whole. They are designed to be broad and all-encompassing so that they can serve any number of uses and scenarios. Essentially, they are lists that are created for general use. This broad nature of general use lists is reflected in the name of the most widely-used word list, West's *General Service List* (1953). Others include Nation's BNC/COCA lists, Browne's *New General Service List* (2014), Brezina and Gablasova's *New General Service List* (2015), and Dang and Webb's *Essential Word List* (Nation, 2016).

Another way of understanding general-use lists is that their objective is to find what is often termed the *core* vocabulary. Though not always explicitly stated, the philosophy behind this approach is that the language being used—usually English—has at its center a self-contained lexicon of essential, primary, basic, fundamental vocabulary that then runs through the entire language. There are layers of frequency and increasing complexity beyond this, with regions of specialized language demarcated for specific purposes such as fields of study or external dialects. Still, this core vocabulary is at the center of it all, and the purpose of a word list is to identify what words fall within its boundaries. Sorell (2013) evaluated a number of definitions of core vocabulary found in the literature. He suggests that general use lists, such as West's GSL, serve as intuitively-selected lists of core written communication, whereas survival vocabulary lists—often found in travel guides or similar materials—are core vocabulary lists of oral communication.

Relatively fewer researchers have created word lists aimed at a more specific purpose or target audience. Specialized-use lists can be designed to only include words that belong to a specific domain, such as a discipline or trade. They can also encompass vocabulary found in a broad range of disciplines, but which are common in a specific context, such as academic texts. In this case, they usually serve as supplements to aid language learners who are already familiar with the core vocabulary of the language.

Perhaps the most well-known example of a specialized-use list is Coxhead's Academic Word List (2000), which replaced the University Word List (Xue & Nation, 1984) as the go-to vocabulary list for aspiring students intent on attending an English-speaking university or those entering the academic world. This is considered a *general* academic word list, since it is for academic use in general, and not for a specific discipline.

More specialized lists include those designed for business English courses, or medical English courses. This is sometimes designated *technical vocabulary.* Nation (2016) explains that technical vocabulary is most often taught after students have mastered general-use vocabulary, and after they have some familiarity with academic vocabulary. Chung and Nation (2003) looked into the nature of a technical vocabulary. By studying specialized words in the fields of anatomy and applied linguistics, they found that a large number of technical words are also found in the language's core vocabulary, or have a general academic use as well. However, when used in a technical text, these words take on a specialized definition that is particular to that domain. This means that much vocabulary is shared across layers of vocabulary, though they may vary semantically, based on context.

2.2.2 Identifying Words (Word Family Levels)

One of the most essential questions that needs to be answered when designing a word list is how one is defining a *word*. Though this may seem like a straight-forward decision, it requires thorough planning and a solid understanding of the theory behind the decision. Should *jump* and *jumped* be counted as two different words or just one? What about irregular inflections such as *go* and *went*? In an article aimed at raising awareness of what he calls the "*Word* dilemma," Gardner (2007) points out that the validity of much vocabulary research hinges "on the various ways that researchers have operationalized the construct of *Word* for counting and analysis purposes" (2007, p. 242).

The literature has generally come to accept some key terms that are helpful when speaking of the way words are counted. Beginning with the most basic measurement and progressing to the most complex, we can choose to count tokens, types, lemmas, or word families.

Measuring *tokens* means simply measuring the total number of words. The sentence "I like small dogs, big dogs, and every other kind of dog" contains twelve tokens—twelve words in total. Counting *types* refers to the number of separate and distinct words. That is, *dog* and *dog* are the same type, but *dogs* is a different type—even a single difference makes them different types. The sentence above is composed of eleven types. A level above this, the *lemma* includes the stem of the word and its inflected forms, but not any derived forms of the word (derived forms are usually considered a different part of speech). So *do*, *does*, and *did* are all the same lemma, but *doable* is not. This is because *doable* has the derivational affix -*able*, which turns it into an adjective. Francis and Kučera define lemma as "a set of lexical forms having the same stem and belonging to the same major word class, differing only in inflection and/or spelling" (1982, p. 1).

Finally, the term *word family* is used to describe an even more inclusive level than the lemma. However, its precise definition has often varied among researchers. Bauer and Nation (1993) sought to rectify this problem through an in-depth classification of English affixes. Borrowing from Thorndike's (1941) study of English suffixes, their grouping was based on a series of eight criteria: frequency, productivity, predictability, regularity of the written form of the base, regularity of the spoken form of the base, regularity of the spelling of the affix, regularity of the spoken form of the affix, and regularity of function. (pp. 255–56) They identified seven "levels" of word families, with each successive one including a larger number of affixes, and therefore a larger number of types per word family. One very useful aspect of their particular system is that it places all the previous levels (type, lemma, etc.) within the same framework. Under their schema, a level 1 word family is the same as a type, a level 2 word family is a lemma (including all regular inflected affixes), and level 7 (the highest level) consists of classical roots and affixes beyond what most speakers any longer consider separate affixes.

Nation himself suggests that for the purposes of language learning, these specific family word levels can be used simply "as a starting point as an initial framework of reference" (2016, p. 36). That is, they are one interpretation of how to systematically count words for a frequency list. These levels are based on criteria that reflect the needs of language learners, rather than on any psycholinguistic theory of how speakers' mental lexicon is arranged. Still, the idea of word families aligns closely with theoretical models that dictate morphological decomposition as a constant. These theories propose that words are often deconstructed into independent morphemes in receptive tasks and recognized that way, for example by deconstructing *jumping* into *jump* and *-ing*. At the other end of the spectrum stand theories that would place *jump* and *jumping* as separate lexical entries (Brysbaert and New, 2009, 982–83).

Either way, there is strong evidence to suggest that inflected/derived forms and their base forms do affect each other in some way, suggesting that word families are a measure of a real representation in speakers' mental lexicon. In one such study, Nagy et al. (1989) explored the effect of both inflectional and derivational family frequency during a lexical decision task. They found that both types of morphological relationships lowered word recognition times, leading to the conclusion that inflections and derivational relationships are both represented in the mental lexicon, either through the grouping of related words under the same entry, or through linked entries. However, all the participants were native English speakers, so to what extent do L2 learners' lexicons reflect the same level of linking?

More recent studies have found that L2 learners' morphological knowledge and word-building ability are not nearly as developed. Ward and Chuenjundaeng (2009) conducted a study that tested the receptive ability of Thai engineering and doctoral students learning English. They were tested for their knowledge of a series of base words, together with various derived forms of the same words. They found a surprising lack of familiarity with the derived words, even when participants knew the base forms from which they were derived. Similarly, but from a productive and not receptive standpoint, Schmitt and Zimmerman (2002) found that learners could produce only a limited number of derived forms when presented with a word family headword. These results challenge the common assumption that "once the base word or even a derived word is known, the recognition of other members of the family requires little or no extra effort" (Bauer and Nation, 1993, p. 253).

There is evidence (Mochizuki and Aizawa, 2000; Schmitt and Meara, 1997) to suggest a positive correlation between vocabulary size and morphological knowledge. If this is the case, then using higher-level word families in Bauer and Nation's framework for word list creation (as is the case in), may not be appropriate for learners with limited knowledge of vocabulary—the very learners that many of these lists target.

Similarly, a study by Jeon (2011) found that L2 learners' morphological knowledge leads to greater reading comprehension. Since many word lists are designed to increase reading comprehension in learners, it follows that they will likely be used by students without strong word-building abilities.

Clearly, then, when it comes to creating a word list, the unit of counting needs to fit the purpose and target audience of that list. Brezina and Gablasova (2015) contend that Bauer and Nation's (1993) higher word family levels ignore the lack of transparency that exists between many of the entries that would be placed under the same word family. Especially when creating a word list for beginners, Brezina and Gablasova point out that the morphological knowledge of language learners is often not developed enough. Because their New General Service List was created for beginners, and since it is intended to aid vocabulary acquisition for both receptive and productive purposes, Brezina and Gablasova chose the lemma as their unit of measure.

Seeking to quantify the effect of choosing to measure word families as opposed to word types, Sorell (2013) compared the text coverage of frequency lists made from the same four corpora. Each corpus corresponded to one of Sorell's text types (see above). Sorell's definition of "word families" was a slightly modified version of Bauer and Nation's (1993) sixth level of affix inclusion. He found, as would be expected, that the most frequent word families have a much larger text coverage than the most frequent types. This is especially true when measuring type coverage—the most frequent word families accounted for roughly 4–6 times as many types in each corpus. However, when measuring overall token coverage, the top word families only covered about 3–10% more than the same number of most frequent types. Sorell also found that the most frequent 1,000 word families consisted of 6,557 word types in the general writing corpus. The number was similar in the other text types, though somewhat lower.

2.2.3 Objective vs. Subjective Design

(Nation 2016:133) >There are two major approaches to making corpus-based word lists. One is to stick strictly to criteria based on range, frequency and dispersion (Brezina & Gablasova, 2015; Dang & Webb, Chapter 15 this volume; Leech, Rayson & Wilson, 2001). The other is to use a similar statistical approach but to adjust the results using other criteria such as ensuring that lexical sets such as numbers, days of the week, months.

Brezina and Gablasova (2015), p. 3: > Seen from the perspective of current corpus linguistic research (cf. McEnery and Hardie 2011), one of the main problems of West's GSL lies in the fact that its compilation involved a number of competing principles that brought a large element of subjectivity into the final product. When reviewing the compilation principles of the GSL, we can see that in addition to the quantitative measure of word frequency, West also used a number of 'qualitative' criteria for the selection of individual lexical items. These include (i) the ease of learning, (ii) necessity, (iii) cover, and (iv) stylistic and emotional neutrality (West

1953: ix–x). Let us now briefly discuss these principles.

2.2.4 Objective Criteria (Frequency, Range, Dispersion)

Nation (2016), p. 103: > Dividing a corpus into sub-corpora allows the creation of range and dispersion figures. In some ways range figures are more important than frequency figures, because a range figure shows how widely used a word is, and this indicates its "general service". Brysbaert and New (2009) found that a range measure was a good predictor of lexical decision times. Carroll, Davies and Richman (1971) found in their study that frequency and their measure of dispersion correlated at .8538 (page xxix), showing that the more widely used a word is, the more likely it is to be frequent. Some words however are frequent in just one or two texts or sub-corpora and may not even occur in others. The use of a range or dispersion figure or both can indicate such words.

Brysbert and New (2009), pp. 984–5: > Another variable that has been proposed as an alternative to WF frequency is the contextual diversity (CD) of a word (Adelman, Brown, & Quesada, 2006). This variable refers to the number of passages (documents) in a corpus containing the word. So, rather than calculating how often a word appeared in the BNC, Adelman et al. measured how many of the 3,144 text samples in the corpus contained the word. They found that the CD measure explained 1%–3% more of the variance in the Elexicon data.

Brezina and Gablasova (105), p. 8: > ARF is a measure that takes into account both the absolute frequency of a lexical item and its distribution in the corpus (Savicky´and Hlava´c ˇ ova´2002; Hlava´c ˇ ova´2006). Thus if a word occurs with a relatively high absolute frequency only in a small number of texts, the ARF will be small (cf. Cerma´k and Kr ˇ en 2005; Kilgarriff 2009). All four wordlists were then sorted according to the ARF that ensured that only words that are frequent in a large variety of texts appeared in the top positions in the wordlists.

Sorell (2013), p. 89: Dispersion.

2.3 Modern Non-English Word Lists

Gardner, D. (2007), p. 242: > Hazenberg and Hulstijn 1996—Dutch language;

3 Methods: Creating the Conversational Hebrew Vocabulary List (CHVL)

3.1 Overview

As we have seen, the brunt of the work in high-quality vocabulary frequency list creation has focused on *English* frequency lists. Outside of the English-speaking world, and especially when dealing with less commonly taught languages, it's difficult to find well-researched word lists, if they exist at all. Why have not more educators—those who may benefit from these lists the most—decided to undertake such a task?

This need not be a project that one starts from scratch every time. Many tools already exist to make the process smoother. Still, with the rapid pace at which technology changes, these tools tend to quickly become obsolete. They are also usually restrictive to the specific preferences of their creators.

Rather than using these tools, I chose to create a series of simple scripts to create the Conversational Hebrew Vocabulary List.

The two most widely-used languages for the type of data analysis involved in a word list creation are Python and R. I chose to use Python for this project. Python was designed specifically to be a very readable programming language. That is, it is easy to read and understand the purpose and flow of the code. This was one of my primary reasons for choosing to use it, since it increases the ease with which this project can be reproduced by other researchers and educators to create their own word lists. R, on the other hand, requires a deeper familiarity with the syntax and conventions of the language in order to understand.

The second characteristic that makes Python ideal for an open-source project of this nature is its mild learning curve. Though considerable effort must be made to learn any programming language, Python is widely considered good for beginners because of its simplicity. With only a rudimentary knowledge of Python, even educators or enthusiasts without a coding background will be able to modify the scripts used here to suit their own needs. To this end, I will also carefully explain what, exactly, the code does.

Though all of the code is included in this thesis (appendix 2), it can also be found in an online repository at https://github.com/juandpinto/opus-lemmas. The repository can easily be cloned, or individual files can be downloaded, for modification and use. The repository uses the version control system *Git*. This means that anyone can easily look through the history of each file to see specific changes that have been made over time.

Suggestions for improvements can also be submitted through the GitHub interface, allowing for a system of cooperation and incremental innovation among researchers. The exported Conversational Hebrew Vocabulary List, in its entirety, can also be found in the repository.

This thesis, then, beyond explaining the theory behind the creation of the CHVL, aims to make the process as reproducible as possible. This section contributes to that aim by carefully documenting each step of the process.

3.2 The corpus

Before coding or analyzing anything, it's important to find an appropriate corpus to use and to become familiar with its structure. A useful place to begin is OPUS, which is part of the Nordic Language Processing Laboratory (NLPL), and hosted by the CSC IT center in Finland. OPUS is a database of many open, parallel corpora. These include corpora of movie and television subtitles, TED talks, web-crawled data, newspapers, and of course, books. The corpora are all free and open to the public.

The CHVL was created using one of OPUS's corpora, the OpenSubtitles2018 corpus. The corpus can be downloaded in a variety of formats, and can be downloaded either as *parallel* corpora, or as a monolingual corpus. A parallel corpus consists of two languages interwoven together. For example, a line from the English subtitles of a movie will be paired with the same line from the French subtitles of the same movie. In theory, this means that each line of the corpus should have the same meaning in two different languages. The creation of parallel corpora has made possible many interesting and useful tools for linguistics, translators, and language learners. These include the open-source CASMACAT project and the ReversoContext tool.

For the purpose of creating a word list, a monolingual corpus is best. Note that parallel corpora will often be composed of less tokens than monolingual ones. This is because parallel corpora will only include movies for which the subtitles exist in both selected languages.

Though it's possible to download plain text files, the most useful format available for download is XML. Indeed, the most common file format used for large corpora is XML. The XML structure allows for nested key-value pairs, which are especially useful for parsed corpora that contain extensive metadata. XML is comparable to JSON, which we will use later to extract specific movie metadata directly from a database.

Another factor to consider is whether to download an untokenized, tokenized, or parsed corpus. An untokenized corpus contains simply the raw lines of text as found in the original subtitle files (divided into lines as they would appear while watching the movie, and labeled with the appropriate time for them to be shown):

```
<s id="49">
  <time id="T39S" value="00:03:22,280" />
 ?      ,
  <time id="T39E" value="00:03:24,120" />
</s>
```

A tokenized corpus has further been split into individual words and punctuation, such that each word is tagged on its own:

```
<s id="49">
  <time id="T39S" value="00:03:22,280" />
  <w id="49.1"> </w>
  <w id="49.2">  </w>
  <w id="49.3">   </w>
  <w id="49.4">,</w>
  <w id="49.5">    </w>
  <w id="49.6">?</w>
  <time id="T39E" value="00:03:24,120" />
</s>
```

A parsed corpus contains much more information for each token. The data included depends on the features of the language and on the parsing script used, but it can include things such as part of speech, syntactic role, lemma, and even specific features like gender, person, and number. Here is an example:

```
<s id="49">
  <time value="00:03:22,280" id="T39S" />
  <w xpos="ADV" head="49.3" feats="PronType=Int" upos="ADV" lemma=" "
      id="49.1" deprel="obj"> </w>
  <w xpos="PRON" head="49.3" feats="Gender=Masc|Number=Sing|Person=2|
      PronType=Prs" upos="PRON" lemma="  " id="49.2" deprel="nsubj">  </w>
  <w xpos="VERB" head="0" feats="Gender=Masc|HebBinyan=PAAL|Number=Sing|
      Person=1,2,3|VerbForm=Part|Voice=Act" upos="VERB" misc="SpaceAfter=No"
      lemma="  " id="49.3" deprel="root">  </w>
  <w xpos="PUNCT" head="49.3" upos="PUNCT" lemma="," id="49.4"
      deprel="punct">,</w>
  <w xpos="NOUN" head="49.3" feats="Gender=Masc|Number=Sing" upos="NOUN"
      misc="SpaceAfter=No" lemma="   " id="49.5" deprel="obj">   </w>
  <w xpos="PUNCT" head="49.3" upos="PUNCT" misc="SpaceAfter=No" lemma="?"
      id="49.6" deprel="punct">?</w>
  <time value="00:03:24,120" id="T39E" />
</s>
```

All of the data used to create the CHVL came from a monolingual parsed corpus of Hebrew. The parsing was all done automatically using .

3.3 Cleaning the corpus

Unlike many corpora, the OpenSubtitles2018 corpus as presented in its downloadable form has already undergone significant cleaning by the OPUS team.(Lison &

Tiedemann, 2016) This is good news, since data cleaning is often the most laborious part of the process. However, there is one issue that must be addressed before the corpus can be used to create a word list.

The files inside the downloaded folder are organized as follows:

```
Zipped folder in GZ format
    Folder for year X
        Folder for movie A
            Zipped XML in GZ format
            Zipped XML in GZ format
            Zipped XML in GZ format
        Folder for movie B
            Zipped XML in GZ format
            Zipped XML in GZ format
    Folder for year Y
        Folder for movie C
            Zipped XML in GZ format
        Folder for movie D
            Zipped XML in GZ format
            Zipped XML in GZ format
            Zipped XML in GZ format
        Folder for movie E
            Zipped XML in GZ format
            Zipped XML in GZ format
    Folder for year Z
          Folder for movie F
              Zipped XML in GZ format
              Zipped XML in GZ format
```

This organization is straight-forward, except for the fact that there are multiple XML files for each movie. The subtitle files that OPUS has collected, parsed, organized, and made available for mass download were all obtained from the Open Subtitles project (hence the name of the corpus). Because this is a database where users can upload the subtitle files they extract from their own movie collection, there are often multiple uploads for the same movie. For our purposes, this results in movies that can have anywhere from a single subtitle file to dozens of them. Unfortunately, though the tokens in the files themselves are usually the same (with only minor variations in the XML metadata), this is not always true. Some few variations seem to be different and independent translations.

Part of cleaning the corpus, then, entails getting rid of these duplicates. As a means of simplifying the entire process, I chose simply to use the first file in each movie folder. I've included the short Python script for this in its entirety in Appendix 3.3. However, I will here explain what it does in detail so that it can be easily modified to fit different circumstances.

The script first makes a copy of the entire folder structure in the original downloaded (and unzipped!) corpus into a new directory. It then finds the first XML file in each movie folder and copies it into the appropriate place in the new folder structure. This means that it doesn't delete or otherwise change the files in the original corpus in any way.

The first block of code imports necessary modules that are used later in the script (`shutil` and `os`). Lines 7 and 8 define where the original corpus is (`source`), and where the new one will be placed (`destination`).

```
4  import shutil
5  import os
6
7  source = '../OpenSubtitles2018_parsed'
8  destination = './OpenSubtitles2018_parsed_single'
```

Next, a single line of code copies all directories and subdirectories into their new location.

```
11 shutil.copytree(source, destination, ignore=shutil.ignore_patterns('*.*'))
```

Lastly, we create a variable that holds all the XML files located in each movie folder, trim the list to just one, and copy that one into its new location. This process is carried out for one movie folder at a time. The originals are left untouched.

```python
for dirName, subdirList, fileList in os.walk(source):
    for fname in fileList:
        if fname == '.DS_Store':
            fileList.remove(fname)
    if len(fileList) > 0:
        del fileList[1:]
        src = dirName + '/' + fileList[0]
        dst = destination + dirName[27:] + '/'
        shutil.copy2(src, dst)
```

With a newly organized version of the corpus, it's now possible to begin the process of reading and processing data. At this stage, I took some time to gather metadata for all the movies in the corpus in order to identify movies that were originally filmed with Hebrew as their primary language (as opposed to translated subtitles). Because I ultimately decided against this approach for the creation of the CHVL, I will skip that step here. However, a description of that entire process can be found in section 4.4.1 - using original-language movies exclusively.

3.4 Reading data

Before calculating any measures such as frequency, individual lemmas must be extracted from the XML files in the downloaded corpus. There are two ways to go about this. Because XML consists of nested tags and key-value pairs, a dedicated XML parsing tool can be used to extract specific information. In this case we would be creating a list of all *values* in the `'lemma'` *key* within each `<w>` *tag*. The value that corresponds to the `'lemma'` tag below for the word   is .

```xml
<w xpos="VERB" head="0" feats="Gender=Masc|HebBinyan=PAAL|Number=Sing|
    Person=1,2,3|VerbForm=Part|Voice=Act" upos="VERB" misc="SpaceAfter=No"
    lemma="  " id="49.3" deprel="root">  </w>
```

A different approach is to use *regular expressions* to search for a specific string of characters and extract every instance of that string. This is a more brute-force approach, since it ignores the structure of the XML file and treats it all simply as raw text. To find a lemma, a very simple regular expression is sufficient: `lemma="[ - ]+"`. This will search for any instance of the characters `lemma="`, followed by a combination of any number of Hebrew letters (at least one), followed by the character `"`.

Despite the existence of various Python modules for parsing XML files, I found a simple search using regular expressions to be more efficient for various reasons. First, not all elements in the parsed corpus contain *lemma* attributes. Second, punctuation and non-Hebrew words are often lemmaticized. This means that even after extracting all the *lemma* values in a file, I would still need to use regular expressions to search through the results and delete any that contain non-Hebrew characters. I chose instead to skip the XML parsing step altogether.

I will now explain the code in the script used to create the CHVL. As with the other code, the entire script in its entirety can be found in the appendix (2.1).

After importing necessary packages and initializing variables, two functions near the beginning of the script serve to open a file and extract a list of lemmas from it.

```python
# Open XML file and read it.
def open_and_read(file_loc):
    with gzip.open(file_loc, 'rt', encoding='utf-8') as f:
        read_data = f.read()
    return read_data
```

```python
# Search for lemmas and add counts to "lemma_by_file_dict{}".
def find_and_count(doc):
    file = str(f)[40:-3]
    match_pattern = re.findall(r'lemma="[ '"+[ -, doc)
    for word in match_pattern:
        if word[7:-1] in lemma_by_file_dict:
            count = lemma_by_file_dict[word[7:-1]].get(file, 0)
            lemma_by_file_dict[word[7:-1]][file] = count + 1
        else:
            lemma_by_file_dict[word[7:-1]] = {}
            lemma_by_file_dict[word[7:-1]][file] = 1
```

We then run both of these functions for each XML file in the corpus directory (defined earlier in `corpus_path`).

```
for dirName, subdirList, fileList in os.walk(corpus_path):
    if len(fileList) > 0:
        f = dirName + '/' + fileList[0]
        find_and_count(open_and_read(f))
```

The `find_and_count()` function finds each instance of the string described above using a regular expression, then adds the Hebrew part of the string—the lemma itself—to a dictionary. The dictionary is named `lemma_by_file_dict`, and its structure looks like this:

```
'lemma': {'path of file': 'frequency of lemma in file'}
```

A dictionary is at its core a list of key:value pairs. Much like an actual dictionary consists of words and their definitions, this dictionary's keys are made up of all the individual lemmas found by our search. For each lemma, the value is another dictionary—making it a nested dictionary, or a dictionary within a dictionary. The keys for each inner dictionary are the paths of all the XML files (movies) that the lemma appears in, and the value of each is an integer that represents how many times that lemma appears in that file (frequency).

After the script reads each file, it returns a complete dictionary. Here is a sample:

```
:' ' }
        '/he/0/5753574/6853341.xml': 168,
        '/he/0/3607000/5764778.xml': 94},
:'  ' }
        '/he/0/5753574/6853341.xml': 3},
:'   ' }
        '/he/0/5753574/6853341.xml': 6,
        '/he/0/3607000/5764778.xml': 2,
        '/he/0/1278351/3777598.xml': 1}
```

Throughout the rest of the script, this nested dictionary serves as the basis for all of the calculations needed.

3.5 Calculations

For each lemma, the CHVL includes three measures: frequency, range, and U_{DP} (dispersion). It uses dispersion as its sorting value. Let's look at how each of these is calculated. Range will be addressed in the export section, since the script calculates it on the spot as the list is created.

3.5.1 Frequency

Since we've already calculated the frequency of each lemma for each individual file, calculating total frequency per lemma is straight forward. The script simply creates a new dictionary, `lemma_totals_dict`, and adds to it every lemma in the corpus as its keys, with the corresponding value being a sum of the frequencies in all files for that lemma. In other words, {'lemma1':'frequency1', 'lemma2':'frequency2', . . . }

```python
for lemma in lemma_by_file_dict:
    lemma_totals_dict[lemma] = sum(lemma_by_file_dict[lemma].values())
```

This returns Using the short example given above, this would result in the following dictionary:

```
' ':262,
'  ':3,
'   ':9
```

3.5.2 $\mathrm{U_{DP}}$ (dispersion)

Dispersion is more complicated. In theory, it should provide a single quantifiable measure that incorporates both frequency and range, and which can then be used to sort the word list. There is no agreed-upon, single way to calculate dispersion, and different researchers will use the words in slightly different contexts. The model of dispersion I have chosen to follow for this project is Gries' dispersion coefficient, or $\mathrm{U_{DP}}$, () calculated from Gries' DP. ()

In order to calculate Gries' DP for $\mathrm{lemma_x}$, we must first make two calculations for each file in the corpus ($\mathrm{file_i}$): the lemma's *expected frequency* if it were perfectly distributed, and its *observed frequency*—or its actual frequency.

$$\textbf{expected frequency} = \frac{tokens\ in\ file_i}{tokens\ in\ corpus}$$

$$\textbf{observed frequency} = \frac{frequency\ of\ lemma_x\ in\ file_i}{frequency\ of\ lemma_x\ in\ corpus}$$

We must then subtract the lemma's observed frequency from its expected frequency, which will return a value between -1 and 1. We can normalize this result by finding the absolute value. Now the closer the result is to 0, the closer that lemma's frequency is in that particular file to what we would expect if it were perfectly distributed throughout the corpus. A higher number (closer to 1), would indicate a heavier load in that file that we would expect.

By performing this calculation for every file in the corpus, adding them all together, and dividing the result by two (since we're using the absolute value and are therefore adding values originally in both directions), we now have Gries' DP. Where `n` is the number of files:

$$\textbf{DP} \;=\; 0.5\sum_{i=1}^{n} \mid \text{expected frequency} \;-\; \text{observed frequency} \mid$$

A DP of 0 represents a perfectly even dispersion, and a DP close to 1 means a more uneven distribution, where fewer files contain a larger load of the lemma's overall frequency. A DP of 1 is not actually possible.

Gries' usage coefficient, or U_{DP}, is an attempt to make this number more useful. DP is first subtracted from 1 and the result is multiplied by the lemma's total frequency. The full equation for U_{DP} is as follows:

$$\left(1 - 0.5\sum_{i=1}^{n}\left|\frac{file_i\ tokens}{total\ tokens} - \frac{frequency_x\ in\ file_i}{total\ frequency_x}\right|\right) \times total\ frequency_x$$

In order to calculate this, the script must first find the number of tokens in each file. Like before, this is done by creating a dictionary, `token_count_dict`, which contains the key:value pairs of file:tokens. Since we already have a dictionary with the number of times that each lemma appears in each file, `lemma_by_file_dict`, we don't need to open and read the files again. Instead, we can add the values in this dictionary and rearrange them into what we want.

```python
for lemma in lemma_by_file_dict:
    for file in lemma_by_file_dict[lemma]:
        token_count_dict[file] = token_count_dict.get(
            file, 0) + lemma_by_file_dict[lemma][file]
```

We also need to know the total number of tokens in the entire corpus. This is a simple matter of adding all the values in the `token_count_dict` dictionary. The final count is saved into an integer variable, `total_tokens_int`.

```python
for file in token_count_dict:
    total_tokens_int = total_tokens_int + token_count_dict.get(file, 0)
```

Finally, the script uses all these measures to calculate DP and then U_{DP} for each lemma, and places them into their respective dictionaries, `lemma_DPs_dict` and `lemma_UDPs_dict`.

```python
# Calculate DPs
for lemma in lemma_by_file_dict.keys():
    for file in lemma_by_file_dict[lemma].keys():
        lemma_DPs_dict[lemma] = lemma_DPs_dict[lemma] + abs(
            (token_count_dict[file] /
             total_tokens_int) -
            (lemma_by_file_dict[lemma][file] /
             lemma_totals_dict[lemma]))
lemma_DPs_dict = {lemma: DP/2 for (lemma, DP) in lemma_DPs_dict.items()}

# Calculate UDPs
lemma_UDPs_dict = {lemma: 1-DP for (lemma, DP) in lemma_DPs_dict.items()}
```

With these values all calculated for each lemma, the only thing left is to sort and create the final list.

3.6 Sort and export

In order to ensure that the words on the list do not have an abnormally high frequency in some subcorpora (movies) and are nearly absent in others, some have suggested setting a minimum range or dispersion. All words that fall below this threshold are discarded, and the remaining words can then be sorted by frequency.

Though this is a more systematic approach than that used to create many early frequency lists, it still depends on a subjective decision and the whim of the researcher.

Rather than setting an arbitrary bar, the CHVL is sorted entirely by Gries' usage coefficient of dispersion (U_{DP}). This *modus operandi* ensures that the order of words itself—not just which words make it onto the list and which don't—is decided by a combination of both relevant measures: frequency and dispersion. This approach also has the added benefit of being entirely objective.

Since we've already calculated the U_{DP} for each lemma, sorting the list is simple.

```python
UDP_sorted_list = [(k, lemma_UDPs_dict[k]) for k in sorted(
    lemma_UDPs_dict, key=lemma_UDPs_dict.__getitem__,
    reverse=True)]
```

A final table is then created (using a list of tuples, `table_list`), with each line consisting of a lemma, its overall frequency, its range, and its U_{DP}. This table is already sorted by U_{DP} as it's being created.

Because the script has not calculated range by this point, it must do so on the spot as it's entering each lemma into the table. It does this with a simple dictionary comprehension that quickly counts the number of files included in the `lemma_by_file_dict`. Here is the resulting code:

```python
for k, v in UDP_sorted_list[:list_size_int]:
    table_list.append((k, lemma_totals_dict[k], sum(
        1 for count in lemma_by_file_dict[k].values() if count > 0),
        v))
```

Lastly, now that everything is organized into a table, the script opens (or creates, if it doesn't yet exist) a CSV file, writes a header line into it (`LEMMA, FREQUENCY, RANGE, UDP`), and exports the entire table into the file. It then closes it to clear the computer's memory cache.

```python
result = open('./export/WordList.csv', 'w')
result.write('LEMMA, FREQUENCY, RANGE, UDP\n')
for i in range(list_size_int):
    result.write(str(table_list[i][0]) + ', ' +
                 str(table_list[i][1]) + ', ' +
                 str(table_list[i][2]) + ', ' +
                    str(table_list[i][3]) + '\n')
result.close()
```

The list is now complete. The next section will explore the list itself more in-depth.

4 The CHVL: A vocabulary list of conversational Modern Hebrew

The Conversational Hebrew Vocabulary List in its entirety can be found as an electronic supplement to this thesis (in CSV format) or at the following GitHub repository: https://github.com/juandpinto/opus-lemmas. A sample of the first 1,000 words is included in Appendix 1.

For discussion purposes, a small sample of the first 20 words is presented in this section.

#	LEMMA	FREQUENCY	RANGE	UDP
1		23446109	43455	0.9480170255915042
2		5638813	43448	0.9420130372643667
3		9850733	43458	0.929266134661147
4		4812778	43450	0.9292364864789281
5		6846782	43426	0.9285176069174289
6		5272808	43433	0.9145688112131216
7		3880654	43439	0.9088900047303463
8		3892328	43445	0.9067041511201389
9		1766990	43430	0.9042865019832009
10		5118759	43441	0.9015544612816044
11		2362419	43403	0.8922532708182579
12		2579370	43420	0.8909904417204713
13		1061614	43411	0.88900672760779
14		1325676	43414	0.8860074112131449
15		1906717	43429	0.8852706380348441
16		1069358	43376	0.8770543442171884
17		839575	43331	0.8668140051895192
18		861163	43321	0.8654587702150129
19		1202416	43323	0.8586088803742931
20		921757	42963	0.8519038846130076
21		799835	43196	0.8515460134208453
22		580549	43306	0.8490225759002181

#	LEMMA	FREQUENCY	RANGE	UDP
23		957476	43311	0.8460669027641473
24		905161	43202	0.8453711530871517
25		519740	43206	0.8426501511461861
26		549346	43192	0.8389916740842122
27		785143	43202	0.8317146818133982
28		585499	43062	0.8311268353322435
29		464852	43120	0.8276119303133131
30		376940	42895	0.8264713920405512

4.1 Organization

4.2 Use

4.3 Expansion

4.4 Challenges and future direction

4.4.1 Using original-language movies exclusively

One of the potential downsides of using the OpenSubtitles2018 corpus is that it includes all subtitles of a specific language, even *translated* subtitles from movies filmed in other languages. The question is, does a translated script represent true conversational language as well as an original script?

This is a question that requires more research in order to answer satisfactorily. Though translated subtitles don't need to try to approximate the length and mouth shapes that a dubbed script does, its quality still largely depends on the skills of a translator. Most importantly, it's possible that a translation will not accurately reflect the register of the original. Again, these are important points to consider.

One solution is to simply use movies that were originally filmed in the target language of the corpus. In theory, each XML file in a monolingual OpenSubtitles2018 file should contain a tag that identifies the original language of the movie. In practice, I found that the overwhelming majority of the files contained an empty `<lang>` tag instead. Luckily, there is a way to obtain the desired metadata for each movie in the corpus.

This can be done with a script that uses an application programming interface (API) to fetch specific information from an online movie database. The name of each movie folder in the corpus, which is simply a series of numbers, corresponds to that movies IMDb ID, which is a unique ID registered with the Internet Movie Database. This makes the process relatively easy, as we simply need to query the database using this ID to receive all of the movie's metadata.

Though IMDb does provide their own API, I decided instead to use an API created for the Open Movie Database (OMDb). This API can be used free-of-charge, but it has a 1,000 movie limit per day. Since the OpenSubtitles2018 Hebrew corpus contains nearly 50,000 movies, I decided instead to pay for a daily limit of 100,000 movies. This only requires a $1.00 donation for each month that one is registered to use the OMDb API.

Once an API key is obtained, a script can be written to obtain the information desired for every movie all at once. In this case, we want to know the original language(s) for each movie.

This script in its entirety is found in Appendix 2.2. It uses an imported Python wrapper for the API, written by Derrick Gilland, which can be found at https://github.com/dgilland/omdb.py. This package can be installed through PIP by entering `pip install omdb` into the command line.

For practical purposes, the script requires one to enter a specific year (or, more accurately, corpus folder name). If desired, an asterisk can act as wildcard: `python OMDb-fetch.py 1988` will fetch data for movies from 1988, while `python OMDb-fetch.py 198*` will do it for all movies in the 1980s. In order to fetch data for all movies in the database at once, use `python OMDb-fetch.py *`. I don't recommend this, however, since it may overload the server and cause the script to time out.

The script begins by creating a list of all movie directory paths for the desired year.

```
15 for name in glob.glob(
16         './OpenSubtitles2018_parsed_single/parsed/he/' + year + '/*/'):
17     IDs.append(name)
```

Each item in the list is then trimmed to include only the name of the movie folder, which is *almost* equivalent to the IMDb ID.

```
20 IDs = [os.path.basename(os.path.dirname(str(i))) for i in IDs]
```

In order to make the IDs match those in the database, additional zeros must be added to the beginning until they are seven digits long.

```
23 for i in IDs:
24     while len(i) < 7:
25         IDs[IDs.index(i)] = '0' + i
26         i = '0' + i
```

The list is then sorted numerically in order to more easily interpret the results: `IDs.sort()`.

The API key is set in line 32, but be sure to replace `906517b3` with your own key, which can be obtained at http://www.omdbapi.com/.

```
32 omdb.set_default('apikey', '906517b3')
```

The script then prints a table header, fetches the title, year, and language(s) for each movie, and prints the results directly into the computer terminal.

```
35 print('# ' + year + '\n' +
36       'IMDb ID\tTitle\tYear\tLanguage(s)')
```

```
        for i in IDs:
            doc = omdb.imdbid('tt' + i)
            print('tt' + i + '\t' +
                  doc['title'] + '\t' +
                  doc['year'] + '\t' +
                  doc['language'])
```

5 Implications for other less commonly taught languages

5.1 Easy reproducibility and growth

Appendix 1: Conversational Hebrew Vocabulary List (CHVL)

LEMMA	FREQUENCY	RANGE	UDP
	23446109	43455	0.9480170255915042
	5638813	43448	0.9420130372643667
	9850733	43458	0.929266134661147
	4812778	43450	0.9292364864789281
	6846782	43426	0.9285176069174289
	5272808	43433	0.9145688112131216
	3880654	43439	0.9088900047303463
	3892328	43445	0.9067041511201389
	1766990	43430	0.9042865019832009
	5118759	43441	0.9015544612816044
	2362419	43403	0.8922532708182579
	2579370	43420	0.8909904417204713
	1061614	43411	0.88900672760779
	1325676	43414	0.8860074112131449
	1906717	43429	0.8852706380348441
	1069358	43376	0.8770543442171884
	839575	43331	0.8668140051895192
	861163	43321	0.8654587702150129
	1202416	43323	0.8586088803742931
	921757	42963	0.8519038846130076
	799835	43196	0.8515460134208453
	580549	43306	0.8490225759002181
	957476	43311	0.8460669027641473
	905161	43202	0.8453711530871517
	519740	43206	0.8426501511461861
	549346	43192	0.8389916740842122
	785143	43202	0.8317146818133982
	585499	43062	0.8311268353322435
	464852	43120	0.8276119303133131
	376940	42895	0.8264713920405512

LEMMA	FREQUENCY	RANGE	UDP
	367902	42714	0.8252849429901923
	397979	43034	0.8227270095370316
	447348	43074	0.8218430306303226
	511696	43109	0.8200130589994714
	424351	42768	0.8199292074719209
	678461	43101	0.8173698432035429
	538120	43151	0.817293896388624
	327641	42552	0.8144761932302511
	548185	43249	0.8123221548952667
	464746	42758	0.8106640851854294
	947724	43291	0.8084645435724982
	490428	43141	0.8064150536835354
	419823	43050	0.8047477721008908
	388089	42849	0.8041853147025249
	321102	42702	0.8041830812885888
	1207533	43226	0.8041799654230262
	433036	42608	0.8024645920670651
	260131	42071	0.80221384969919
	394738	42700	0.8014874792904632
	373446	42688	0.8007373599813871
	255588	41924	0.7996141927734344
	310681	42564	0.7980211954070152
	270578	42075	0.7975860019513064
	286598	42191	0.7952189434137065
	562845	42958	0.7942488823605546
	412974	42796	0.7936468503374203
	267993	41984	0.791768039476287
	218184	41190	0.7917160839073898
	205086	41000	0.7894097507718038
	289788	41648	0.7883758931999164
	214954	41188	0.7877135037513165
	225776	41249	0.7876190346775811

LEMMA	FREQUENCY	RANGE	UDP
	250376	41870	0.7861533935879315
	316008	42239	0.7851973471872901
	277379	41924	0.7851063904438749
	298579	42161	0.7839807768535207
	208160	40811	0.7825677023294968
	260354	42041	0.782461191312035
	201709	40669	0.7812562648331688
	178461	40099	0.7787652199040754
	224000	40829	0.7783418455112182
	181166	40252	0.7771570268839243
	238416	40919	0.7759852575633128
	139866	38453	0.7743368394415071
	623430	41759	0.7739847609912638
	171278	39499	0.772953152626896
	172041	39452	0.772604755682636
	158697	38893	0.7723833042726347
	157377	39393	0.7718752651047299
	154089	38931	0.7716276663855711
	306213	41152	0.770282592043259
	154130	39146	0.7695547607230302
	198165	40314	0.7680001789230777
	202999	40579	0.7671689493296213
	162483	39369	0.766231651775358
	99971	35015	0.7652934349176803
	141725	38426	0.765223647162116
	55050	27901	0.7643551071644706
	266260	41382	0.7642633606613978
	75388	31940	0.764149835626775
	95518	34110	0.7641391655827925
	279723	41591	0.7624500121440642
	138136	37831	0.7622924022444642
	54161	26863	0.7622073427780777

LEMMA	FREQUENCY	RANGE	UDP
	168829	39248	0.762146187601928
	256770	41514	0.7618890654783477
	138710	37943	0.7616653193582906
	109842	35652	0.7609634126087095
	100832	34923	0.7608985491551321
	183652	34316	0.7606625397457291
	46736	25727	0.7606528159744843
	54522	27109	0.7599555507835318
	71584	31900	0.7598150176879501
	220597	40784	0.7597664595282851
	74650	30977	0.7595020775107555
	271131	40994	0.7593203248667084
	117845	36660	0.7590572559633229
	43631	24044	0.7586111453330702
	65759	29695	0.7585663791048742
	117371	34092	0.7581777014808021
	115785	36237	0.7580135333579686
	64765	29164	0.7578402237480675
	81313	31883	0.7575578844994758
	63374	28020	0.757521047413982
	140013	37324	0.7572790956219879
	48054	25189	0.7572739996587984
	179147	39606	0.7570976823525724
	64559	29600	0.7570520698232999
	52040	26619	0.7569118990722683
	73740	31360	0.7568256123829105
	175325	38554	0.7568215267026179
	57163	27476	0.7568073870178758
	50853	25409	0.7565141918687514
	103159	34456	0.7563573904210092
	146976	38291	0.7559666453540768
	70535	29954	0.7559403174686339

LEMMA	FREQUENCY	RANGE	UDP
	90712	32860	0.7559332038697861
	131321	36335	0.7556745825603102
	106181	34201	0.7553535119497476
	70421	30093	0.7553442662313572
	52191	25779	0.7550953779006937
	115301	35648	0.7549545996691428
	69814	30269	0.754847028892198
	90429	33029	0.7548084431901666
	87778	32785	0.7547254039813581
	50540	25479	0.7545120898881357
	46983	24902	0.7544822115365516
	89700	33337	0.7536895580188299
	44150	24171	0.7534149166420128
	59244	27026	0.7527902619292293
	60990	27698	0.7527396447680208
	233644	38074	0.7527022740569967
	53987	25929	0.7525384927223415
	146411	36788	0.7521875367283136
	110859	35168	0.7520426531036938
	64442	28808	0.7520203664772012
	40967	23297	0.7519715819024961
	56953	26975	0.7516978166931896
	46255	24614	0.751597228249568
	49967	25437	0.7515551797694072
	45437	23959	0.7511920906615481
	46761	24239	0.7509541013859027
	79758	31121	0.7509185448215709
	119322	35370	0.7509148800205059
	98287	33880	0.750598796653489
	133888	37080	0.7502095967056479
	54812	26160	0.7502076842128453
	72821	29977	0.7500879198937482

LEMMA	FREQUENCY	RANGE	UDP
	243889	40776	0.7500171150896289
	45291	23369	0.7499018221667755
	79634	30810	0.7497801544409235
	51586	25538	0.7494769452339677
	166920	38806	0.7494677516396457
	133192	36380	0.7493763190047982
	41990	22842	0.7493114347723449
	182308	39208	0.7490466276444374
	42481	23367	0.7489349234567368
	39467	22636	0.7484708894038101
	57882	27334	0.7484672402261972
	219155	39679	0.7479740349720239
	41737	22893	0.7479543670013787
	71177	28968	0.7478055485649457
	48992	24559	0.7475294553035913
	87864	31625	0.7475109849142201
	269458	40779	0.7473604624367596
	80911	30543	0.7471789501721844
	49393	25267	0.7469436120618338
	100464	33988	0.7468575081363968
	76694	30709	0.7457989917185035
	43162	23299	0.7452825579352562
	161107	37991	0.7452401041821652
	82463	31103	0.7451788216108366
	45787	23109	0.7450939608395166
	79849	30971	0.7450116709435397
	55858	25677	0.7449420522970438
	49463	24500	0.7448628880292192
	55491	25902	0.7448479538853883
	59141	26640	0.7448047005939695
	126600	35753	0.7444971437090584
	63422	26925	0.7441921309235029

LEMMA	FREQUENCY	RANGE	UDP
	61827	27273	0.7441448175833947
	242051	40437	0.744128257691816
	42945	22239	0.7436579638965237
	115658	34716	0.7435984845608715
	48731	24396	0.743572883671449
	77133	28830	0.7435419137850303
	47299	23392	0.7435258737990725
	148715	37651	0.7433888266022728
	60631	26575	0.7430392985230476
	638998	43040	0.7429720343179576
	39982	21679	0.7427308034801754
	834217	42733	0.7426789343012317
	54710	25545	0.7425742773006223
	68827	28544	0.7423705866199584
	45562	23096	0.742354884698041
	84165	31396	0.742233925433707
	34689	20798	0.7422119424218817
	190553	38552	0.7421879954381134
	142727	37387	0.7420182822169226
	48373	23811	0.7419727546546413
	44182	22621	0.7419283550397897
	152422	35355	0.7417440599483898
	121973	34939	0.741717091321128
	98210	32986	0.741396414315947
	58550	26144	0.7413615068156847
	85029	31038	0.741264867560074
	49030	24642	0.7410606074082611
	62874	27460	0.7408563334636658
	36919	21077	0.7407312507161847
	83000	31491	0.7407147298634207
	42643	22444	0.7404669741689867
	39596	21409	0.7402443895778394

LEMMA	FREQUENCY	RANGE	UDP
	92062	30773	0.7400989552531372
	42332	23031	0.7400542462575641
	55083	25663	0.7399883644877214
	82840	31018	0.7395963524477951
	45868	22666	0.739459861280424
	67213	27781	0.7393956671105517
	76011	29258	0.7393012974629778
	42863	22750	0.7389565606452119
	77021	29885	0.7388757029381893
	124636	35618	0.7386129576435905
	37443	20788	0.7385213779401002
	162906	37277	0.7383491240296027
	43011	21960	0.7382879283282054
	105627	34354	0.7381245585750493
	199458	38203	0.7381112110810331
	206740	39632	0.738090718047284
	56416	25495	0.7379336542433617
	54321	24952	0.7377121059648806
	43581	22423	0.7377085338439194
	176643	38711	0.737634347934375
	45180	22991	0.7375918660462768
	71132	28630	0.7375786819664074
	50618	23796	0.7373942766665277
	58403	24849	0.7372285978500577
	57852	25977	0.7372138662294929
	70052	27659	0.7371124211213822
	93067	31268	0.7368065448570824
	38191	21002	0.7367866744970561
	79627	29256	0.7363298635187498
	76463	28622	0.7362924936206932
	44598	23024	0.7362295075600527
	47729	27236	0.7362027981454538

LEMMA	FREQUENCY	RANGE	UDP
	44487	22043	0.7358839517441256
	38921	21239	0.7358829875145043
	32456	20251	0.7357807778360249
	36934	20718	0.7353728441350172
	58792	25572	0.7352874903624752
	77624	30078	0.7352431471461518
	83352	29667	0.7351487295387624
	39769	21157	0.735108190645487
	65515	26896	0.7350893805265912
	41951	22067	0.7349590884128316
	44400	22438	0.734678475903399
	59142	25896	0.7344972775963964
	186844	38452	0.7341477279708515
	33427	19982	0.7341361878703434
	48709	23158	0.7339458687101879
	57056	24733	0.7337693895037696
	58804	25849	0.7337249196449549
	267134	40244	0.7336376602389013
	37336	20241	0.7335996356533924
	36998	21089	0.7334231511678592
	112118	34293	0.733336991457137
	47572	23466	0.7333269908370075
	73858	28723	0.7330549250284261
	51076	22791	0.7330223661798917
	43425	21982	0.732873446177092
	33666	20052	0.7327245221691426
	55246	24773	0.7326923297385193
	179543	37349	0.7323951341056578
	63071	25269	0.7323940253520271
	120923	34685	0.7323215002796702
	327260	40888	0.7321608086979936
	148724	36977	0.73188325905325

LEMMA	FREQUENCY	RANGE	UDP
	44157	21790	0.7318474783586004
	111812	33563	0.7316673429471234
	60763	26180	0.731573766965979
	56993	24951	0.7315141466948976
	111934	33647	0.7315138676614863
	146801	36266	0.7315131650859238
	38500	20700	0.7314891949830211
	83284	29329	0.7312992409688314
	65270	27051	0.731232327828709
	46087	21743	0.7312303244118035
	276544	40438	0.7309931604682163
	134031	35660	0.7308407835483648
	81307	29720	0.7305176276542591
	100316	31667	0.7301189214517326
	48959	22603	0.7298625379735366
	92028	31259	0.7297169818232159
	41876	20973	0.7296060877686469
	104109	32606	0.7295537700356013
	43823	21207	0.7295246894541164
	32762	19333	0.7294969843193397
	43930	21609	0.7294567962569831
	33532	19572	0.7292495449894719
	113796	34419	0.7291835982126904
	92936	31255	0.7291680677639987
	51291	23015	0.7290616868435258
	135096	35138	0.7288138227784254
	44903	21657	0.7285867445795255
	159524	36882	0.7285685947459715
	99172	30693	0.7281264133078738
	49933	22825	0.7281093816402362
	37067	19944	0.7280241904637095
	53850	23964	0.728006820808258

LEMMA	FREQUENCY	RANGE	UDP
	36698	20032	0.7278323203778543
	60733	25833	0.7277972419374569
	35553	19488	0.7277121855754171
	141977	35703	0.7276218143103719
	91416	29969	0.7268211844742474
	45480	21633	0.7266399145635196
	78950	29281	0.7265271334180008
	42718	21524	0.7263799242667838
	81780	30119	0.7262683134873333
	38008	20056	0.7261426773923043
	48123	22479	0.7259277175450743
	121582	33952	0.7253824561031772
	63828	25651	0.7252015380149588
	51446	22382	0.7251261161094453
	51727	22837	0.7249730966533403
	183452	38124	0.724901564808779
	46982	21498	0.7248317713528625
	88518	28965	0.7245400688505927
	44407	20843	0.7244146558563695
	71098	25533	0.7242354424500794
	33998	19445	0.7241870178351418
	68431	25592	0.7240913471563389
	28977	18426	0.7240666724040008
	458405	41920	0.7238993222674358
	44367	20845	0.7238395600699958
	34301	19572	0.7237249770504071
	29358	18330	0.7236610722630771
	35412	19710	0.7236317511261257
	33710	18756	0.7235971022375047
	36007	20104	0.7235659300684095
	77454	24793	0.7234881969545905
	55784	23190	0.7231869588549322

LEMMA	FREQUENCY	RANGE	UDP
	32794	18770	0.7228958514266373
	38827	19897	0.7228810085143214
	51801	22795	0.7227687501012691
	277322	38980	0.7226128672546543
	33218	18888	0.7221544319255918
	28974	18294	0.7219771835548576
	209263	39030	0.7219628991896636
	32277	19032	0.7219402361903444
	34626	19226	0.7218224720693467
	38503	19735	0.7218197451981162
	68492	25911	0.7216722253439476
	31904	19026	0.7213861778901041
	38847	19906	0.7213047460203668
	40865	19918	0.7200534594753326
	45515	21452	0.7199803256628934
	57377	23196	0.7199797507330308
	72795	25912	0.7198548977852953
	76549	27783	0.7197942704874101
	38896	19160	0.7193984301156588
	28195	17633	0.7179471354606775
	31938	17993	0.7178875479291318
	39722	19976	0.7178318885948425
	30611	18112	0.7175492795448047
	169318	35873	0.7173176144571274
	34966	18622	0.7171228943133598
	28968	17836	0.7170220434687311
	30214	17975	0.7167067263462843
	389942	38399	0.7164040078175146
	42114	20366	0.715742044595925
	35529	18869	0.715623873615767
	32852	18273	0.7153206863232138
	92894	28956	0.7152910961863745

LEMMA	FREQUENCY	RANGE	UDP
	25388	17308	0.715242147762549
	46925	21649	0.7152074257253475
	55510	22331	0.7150010542855055
	61731	24590	0.7148350498423748
	39306	19993	0.714780566876325
	28205	17294	0.7147342762897959
	69616	26111	0.7147277465249305
	109017	31987	0.714111894690228
	51623	21817	0.7139722357520653
	45566	21069	0.7135320107753302
	48561	20788	0.7134925247145726
	72672	25859	0.7134712581772495
	63139	23852	0.7133698597353025
	46518	20269	0.7133580286170553
	55806	23413	0.7131913761368781
	29234	18039	0.7130681253147519
	38658	18630	0.7128121408958408
	60595	24076	0.7127045890658317
	32161	18648	0.7126869215118357
	32783	18349	0.7124880102435501
	26211	17189	0.7124481145857819
	58699	23577	0.7120624871561572
	30242	17658	0.712024476667253
	200100	38089	0.7118554468204595
	31511	17589	0.7113908312168006
	31149	17418	0.7112366987231573
	190693	37369	0.7108608251852613
	77117	26285	0.7107238238394067
	38107	18684	0.710590838415617
	28613	17301	0.7105602371787167
	40371	19338	0.7105099410416833
	42011	19849	0.7098332821338336

LEMMA	FREQUENCY	RANGE	UDP
	33585	17992	0.7097300266106569
	30663	17127	0.7082659531348905
	35455	18292	0.7078307699205377
	26957	17337	0.7076901203302396
	27779	16623	0.7075250816982701
	28859	16944	0.7074280407222893
	30722	17521	0.7074048669605393
	35600	19872	0.707384836436548
	28375	16743	0.7072367129840407
	30064	17246	0.7071152849829694
	51262	20354	0.7064758116677619
	106212	31680	0.7063031666204529
	88493	28003	0.7057294006375665
	134482	34158	0.7054149329934163
	26666	16544	0.7054148652184558
	27833	16806	0.7052770346020572
	25711	16436	0.7046718770583188
	31478	17318	0.7046126397785415
	31315	17232	0.704381187050183
	38263	18486	0.7037014242698807
	47853	20598	0.7036041426071771
	112610	31444	0.7035715970714231
	77143	26625	0.7034012996805629
	43784	19665	0.703382716560681
	68326	23615	0.7032628794004314
	52029	20845	0.7030524609907893
	22880	15934	0.7028798910814902
	67543	24487	0.7028664819672402
	26705	16236	0.7028281893587887
	37657	17985	0.7025931732203696
	23722	15917	0.7025409908939945
	27256	16600	0.702504349351031

LEMMA	FREQUENCY	RANGE	UDP
	33426	17264	0.7024540764130105
	68624	24177	0.7023060468215147
	66712	22700	0.7022575358245418
	34085	17830	0.702185171265955
	24649	16114	0.7020679480087713
	24328	15835	0.7020622143011005
	27809	16712	0.7018327426089931
	26092	16054	0.7015122967605076
	85841	24695	0.7007542386170791
	270834	40029	0.7002527794347495
	31729	16994	0.700177473942825
	28455	16294	0.7000108207105036
	30267	16942	0.6996247424104134
	28007	16353	0.6995253096954483
	44028	19759	0.6992853963980246
	27482	16304	0.6992744886973036
	26167	16043	0.6992328040535931
	23209	15773	0.6991071693027469
	29262	16540	0.6989779600488024
	29227	16438	0.6989031230275972
	31504	16983	0.6985069201563415
	29275	16629	0.6984565371973304
	25278	15635	0.6979374643218452
	93599	28383	0.6975583872136064
	22176	15147	0.6971721905273025
	27450	16186	0.6971600738948582
	21710	15161	0.6967019141134994
	39941	17507	0.6963750102399013
	104525	29823	0.6963376645918169
	45069	18671	0.6961500604806219
	210842	35618	0.6959731886044236
	22297	15089	0.6959566077889343

LEMMA	FREQUENCY	RANGE	UDP
	100331	28613	0.6958195850013711
	22495	15287	0.6955729046177865
	28604	16103	0.6955235509773963
	29549	16451	0.6955129842759566
	28762	15978	0.6955030906358723
	28681	16105	0.695348937280591
	32800	16692	0.6952133673277884
	38738	18075	0.694836718083059
	119470	31398	0.6946615752362186
	22878	15250	0.6942184777046629
	24446	15259	0.6941596046712281
	30970	16257	0.6939727559315096
	38433	17001	0.693809383722765
	62249	22129	0.6937258948067699
	36703	17273	0.6935358098044206
	53856	19641	0.6933660069353219
	22521	14991	0.693262684560845
	45814	18981	0.6930330541166263
	24793	15183	0.6929591896222271
	25446	15526	0.6929320519713572
	29266	15645	0.6927257115337238
	26467	15580	0.6923941154883937
	23039	15104	0.6917846176668808
	21524	14882	0.6914479498929901
	126021	32397	0.6911662597780857
	27210	15503	0.6908194406338778
	31794	16392	0.6900902886082496
	24740	15311	0.690039032257495
	22094	14770	0.6900377888467358
	26690	15608	0.6900352932746843
	22210	14629	0.6897565967300394
	24566	15334	0.6895847836301989

LEMMA	FREQUENCY	RANGE	UDP
	31053	15961	0.6895422947125525
	28723	15767	0.6894819474753976
	26512	15122	0.6892665452386049
	33808	16076	0.6891480192237379
	32953	16491	0.6889814461893508
	43516	17953	0.6888180191827749
	42280	17816	0.688741541741356
	25677	15380	0.6887395018052178
	30626	15537	0.6885656229330432
	25563	14887	0.6884474358469187
	21223	14588	0.6883200942315169
	23630	14878	0.6882646629587634
	60204	21244	0.6882338866032024
	23693	14809	0.6881210183299618
	28408	15699	0.6879757716237938
	25377	14927	0.687906105141351
	46803	18838	0.6878891266177269
	84067	26088	0.6878717009477358
	24555	15123	0.6876706830908542
	388785	36676	0.6876699482477497
	22289	15141	0.687650720706058
	22008	14641	0.6874974022012977
	22200	14506	0.6874949659827634
	29006	15672	0.6874567503163083
	24841	14984	0.6873558060561966
	30612	15835	0.6872947395692212
	24282	14940	0.6872409139282645
	21780	14581	0.6870485964716475
	60182	21080	0.6870229797469324
	26786	15514	0.6866285704286212
	65992	23354	0.686503525161127
	44828	18112	0.6864818853801662

LEMMA	FREQUENCY	RANGE	UDP
	26796	14963	0.6863061150233892
	21610	14377	0.6860225856036907
	34611	16310	0.6859805941723799
	33106	15951	0.6859462123211175
	23950	14506	0.6854226527125867
	22066	14658	0.6852878130208068
	21051	14105	0.6852823892765749
	25170	14682	0.6850446951375537
	28113	15212	0.68495638859173
	25315	14298	0.6845052353557717
	48349	18292	0.6842906115290424
	286526	39003	0.6842788489087734
	21503	13788	0.6842106249859392
	29494	15370	0.6841469783002849
	48434	18007	0.6839113203435216
	22093	14170	0.6838908591629407
	22057	14384	0.6837318084442338
	19504	14490	0.6835461455993526
	39099	15768	0.6835264385319766
	24286	14787	0.6834645922664416
	22733	14323	0.6831493689129768
	36237	16121	0.6830663561103945
	25428	14256	0.6829699096633608
	42567	16988	0.6826894040931465
	21816	14095	0.6826519838205198
	27115	15146	0.682634243503716
	37642	16139	0.6826077033912948
	21309	14788	0.6821787424599526
	25497	14268	0.6813941773057559
	24481	14247	0.6812393256781784
	19994	13961	0.6808298314996895
	29391	14943	0.6806467382966965

LEMMA	FREQUENCY	RANGE	UDP
	31627	15003	0.6806137431213461
	21644	14144	0.6804024728140516
	25313	13675	0.6803961451653433
	33414	15815	0.6802982857559461
	79895	21603	0.6801781522494572
	21069	14108	0.6800283771911733
	58850	20345	0.6798777039146211
	23986	14088	0.6798187118264024
	24270	14421	0.6796423144108295
	24742	14245	0.6790634589542364
	22824	13761	0.6788524481662532
	20802	13651	0.6787799978154678
	20797	14047	0.678750450463812
	30675	15261	0.6786662669265437
	21785	13954	0.67864758391202
	23551	13857	0.6782422631264036
	129020	29925	0.67821856494834
	23236	14037	0.6781059413858146
	22585	14040	0.6780237312320938
	21537	14270	0.6779511674009746
	26079	14048	0.6776427473788476
	18128	13540	0.6775737869109377
	27799	14494	0.6775318892418898
	19667	13377	0.677064829028893
	28881	14657	0.6767060865278618
	24189	13885	0.676610398793255
	26088	14117	0.6766056949484114
	20917	13586	0.6763634771906497
	20911	13562	0.6763225833722596
	19466	13467	0.6760658873317587
	23856	13940	0.6759811965992149
	20937	13702	0.6758237864149625

LEMMA	FREQUENCY	RANGE	UDP
	21915	13987	0.6758127032466508
	21314	13618	0.6755505568540574
	21587	13941	0.6755314780968456
	23977	13697	0.6755251882291439
	20336	13425	0.6752554862067313
	64140	20248	0.675181337122335
	60965	18338	0.6751702351525706
	21790	13717	0.6749022499692857
	23118	13684	0.6747570406970882
	18514	13358	0.6747451014554795
	25562	14499	0.6747042682670725
	40225	15443	0.6746088468153335
	31646	14360	0.6745423266599693
	22389	13661	0.6744828553459574
	27716	14014	0.674422893545376
	35362	15119	0.6742473556556765
	21545	13581	0.6736649998327645
	22388	13556	0.673518266896731
	19826	12930	0.6730627154978392
	62585	18340	0.6727316176836081
	27135	13873	0.6726623337497523
	19926	13289	0.672412299984086
	21128	16314	0.6723301941034011
	45488	16659	0.6720249319930169
	23577	13216	0.6719295774727104
	20353	13421	0.6718465453781812
	29533	14189	0.6709086698187956
	34139	14898	0.6708326019578906
	43491	15589	0.6708308058570287
	20903	13155	0.6705956002732566
	29702	13967	0.670526338091348
	18209	13342	0.6704775286241207

LEMMA	FREQUENCY	RANGE	UDP
	35547	14565	0.6703819782869351
	40544	15813	0.670310873278672
	22918	13256	0.670228542095801
	21146	13207	0.6701005456853044
	21596	13566	0.6694625760407247
	19147	12809	0.6693526876419557
	18269	13339	0.6692873399665265
	25216	13233	0.6692700889409678
	19687	13220	0.6690227782342067
	21693	13107	0.6689513536144345
	26687	13752	0.6687970944910632
	78708	21273	0.6686906036115429
	17646	12447	0.6686623456697935
	40008	15669	0.6686397383537792
	19235	13055	0.6685594793070926
	62041	20297	0.6685565848276945
	21093	13101	0.6685136520256576
	25538	14072	0.6682454931711574
	22701	13352	0.6682395192859057
	17831	12680	0.6679289203272101
	21269	13108	0.6678991286359033
	24942	13316	0.66786279804284
	20702	12815	0.6676860514595124
	44899	16449	0.6674228528279049
	19160	12787	0.6673955164452664
	20060	13077	0.6673158968480442
	18684	12827	0.6671985811374572
	42679	15685	0.6670326267165381
	19901	12818	0.6670280318080888
	135843	28087	0.6670047227709521
	26386	13403	0.6668214889402377
	20520	12665	0.6666343952453573

LEMMA	FREQUENCY	RANGE	UDP
	37360	15071	0.6665901442596323
	25354	13620	0.6661673580106098
	23703	13087	0.6659784685998957
	22812	12825	0.6658906288786375
	25814	13385	0.6655604852090624
	28281	13386	0.6655441795191716
	20248	12484	0.6654212333412851
	29409	13552	0.665388921689057
	81151	24065	0.6653530878621438
	36739	14271	0.6653430249451456
	22207	12759	0.66528986685753
	19709	12705	0.6651821943160653
	18666	12929	0.6649849947847231
	27340	13261	0.6646339454080104
	20069	12874	0.6640737606854539
	48255	16013	0.6640217226995406
	18140	12666	0.6639917196295132
	24762	13129	0.6639182069973847
	24612	13205	0.6638577068831049
	18082	12577	0.6637210852906378
	19290	12506	0.6634580464952684
	43989	15605	0.6634340093390043
	18686	12460	0.663249757731597
	29162	13749	0.6632355501650896
	19924	12307	0.6630216875258338
	17813	12195	0.6629976725980056
	56166	15772	0.6623824348949969
	15921	12093	0.6622270427319024
	18260	12276	0.661663178164439
	25313	12838	0.6616223305582061
	22191	12523	0.6606712045749353
	17872	12496	0.6604553639090496

LEMMA	FREQUENCY	RANGE	UDP
	20866	12297	0.6601842499661914
	19558	12295	0.6597971877805462
	18444	11980	0.659273865870732
	18628	12296	0.6591882722249036
	17128	12116	0.6590920682535741
	27010	12727	0.6590021388864132
	22819	12431	0.6589085092110436
	20460	12361	0.6588475702251767
	188842	27119	0.6588334999518092
	18466	11999	0.6588048374229982
	17959	12138	0.6586872275932699
	32434	12971	0.658604770393032
	30789	13232	0.6582379080959757
	19112	12338	0.6581687604873252
	17162	12090	0.6580893063018656
	17794	11925	0.6577512016533622
	21741	12174	0.6575986185167797
	18726	11840	0.657580000764588
	16502	11762	0.6572396301336128
	15521	11779	0.6570854370200072
	18670	11951	0.6568365933585736
	23272	12398	0.6565952224152927
	17282	11976	0.6564856378989506
	19090	11630	0.6564817287664954
	23858	12452	0.6562881351130655
	22920	12078	0.6561303902821363
	28440	12785	0.6558061860300689
	25012	12676	0.6558036879241833
	43969	14317	0.6555802175280518
	20145	11950	0.6554877140836032
	16851	11850	0.6551502507457252
	31147	13077	0.6551130059442934

LEMMA	FREQUENCY	RANGE	UDP
	17331	11866	0.6550067757604334
	20836	12112	0.6549937170192437
	43193	14580	0.6549903666021402
	22583	12072	0.6544375918097031
	18451	11900	0.6543854145932011
	23426	11969	0.6543603975998784
	18549	12031	0.6540224020948957
	18311	11455	0.6540123151736245
	29916	12270	0.653985097002477
	25747	12378	0.6538947506027191
	52329	16423	0.6537311197023931
	17298	11673	0.653518664951362
	198854	34380	0.6534927601094012
	18212	11751	0.652994842218859
	17267	11626	0.6528978377282928
	43138	14561	0.6528336538050364
	18922	11831	0.6528197914372252
	22331	12188	0.6527805513648037
	17187	11732	0.6526623736299149
	16225	11795	0.6526518281726872
	23205	12092	0.6526375596276416
	26021	12524	0.652495110265553
	158901	29216	0.652422065293693
	20138	11698	0.6520770683839752
	18556	11541	0.6517396914339035
	15428	11528	0.6516175159931119
	20356	11784	0.6514161345018492
	18483	11350	0.6513389502713426
	39156	13243	0.6513068634869992
	16495	11512	0.6512381499216253
	18850	11749	0.65123314144539
	18391	11674	0.6511786639194337

LEMMA	FREQUENCY	RANGE	UDP
	23175	12246	0.6510946499325851
	17792	11568	0.651089430095152
	18936	11502	0.6509383997151734
	179198	32392	0.6507953679661866
	17128	11360	0.6507900753644843
	19129	11414	0.6507829235205977
	19177	11500	0.650697790840947
	63906	17859	0.6505429884091367
	14728	11185	0.6502090406030819
	167065	26720	0.6500226526765451
	14318	11272	0.650017478199129
	19265	11584	0.6499870609401818
	43252	14172	0.6499634778612149
	18688	11152	0.6496699478460655
	14313	11145	0.6493888189224581
	60144	16162	0.6493052612958472
	16890	11356	0.6491876543680684
	18472	11298	0.6491741908324218
	146704	26570	0.649022412921918
	15857	11352	0.6488646481768179
	22104	11679	0.6487376366145082
	34636	12766	0.6486768103679672
	17474	11527	0.6486502521988506
	25611	12259	0.6486182909840161
	22502	11556	0.6485146259216399
	15215	11122	0.6484347420644337
	18191	11034	0.6483853237719011
	16442	11215	0.6479885944425221
	20113	11235	0.6478501143692195
	17573	11244	0.6476732706195836
	24456	11606	0.6471709214906182
	14982	11125	0.6471310955755603

LEMMA	FREQUENCY	RANGE	UDP
	37332	12801	0.6470783704276308
	16651	11046	0.6468596260696822
	14287	11109	0.6466812121619336
	15714	11067	0.6466228783089423
	39948	12963	0.646568129713009
	18661	11038	0.6463500222622951
	15711	10810	0.6460287957945534
	17476	11205	0.6458046389907282
	18824	10975	0.6454430352163179
	15694	10759	0.6450294634970144
	56235	14924	0.6448564384336954
	15478	10674	0.6443406094518724
	26116	10942	0.6440801889822201
	16005	11043	0.6440009209798243
	17343	10900	0.643941639114771
	19237	10929	0.6436605734488388
	16549	10838	0.6435246041297097
	14837	10636	0.6434463894608717
	16261	10768	0.6431411339137254
	14300	10511	0.6431050256286064
	14436	10726	0.643083273616774
	17056	10579	0.6428473990487986
	23225	11155	0.6427641540707181
	16150	10716	0.6426654686756792
	14024	10760	0.6425963739155085
	22675	11340	0.6425716518593663
	14943	10646	0.6424313061546942
	18094	10857	0.642349846228184
	49123	14886	0.642154949189438
	14541	10781	0.6416158381534185
	14601	10783	0.6415822338258697
	17003	10723	0.6415436782950756

LEMMA	FREQUENCY	RANGE	UDP
	15086	10705	0.6413020834984038
	14788	10525	0.6412690056547307
	16928	10523	0.6411804032271053
	13603	10314	0.6411760238005964
	24245	11215	0.6411259745670259
	13857	10689	0.6409341156465047
	14841	10993	0.6409186483527978
	15394	10563	0.6408412390143203
	15213	10469	0.6405967234758672
	21259	10718	0.6405054386934129
	33861	12372	0.6404002899811362
	14270	10610	0.6403974806106771
	17663	10886	0.6402669368146068
	17084	10755	0.6401895042409349
	16945	10677	0.6401645513713867
	17203	10547	0.6401587224140988
	17729	10483	0.6401000298544919
	15614	10431	0.6400960609720269
	97424	22002	0.6400922382242324
	16194	10766	0.6399348394341203
	14931	10624	0.6397692213608978
	14399	10453	0.6397504879382999
	15599	10380	0.6396627982991612
	15781	10636	0.6395637872950573
	15101	10488	0.6395623344833508
	16258	10662	0.6394894516169294
	119659	23260	0.6394781787568173
	31706	11132	0.6394041344466985
	14931	10948	0.6392676480721704
	26783	11409	0.6392602266060039
	15348	10445	0.6390752100033101
	21154	10912	0.6388078306657496

LEMMA	FREQUENCY	RANGE	UDP
	20255	10730	0.6386307043411037
	18504	10630	0.6382189051633316
	17929	10591	0.6381770052829154
	16401	10783	0.6381297552224408
	21061	10556	0.6381147360687489
	13449	10338	0.6381142232984156
	34311	12007	0.6380696722065444
	15444	10331	0.6375348013613042
	14132	10389	0.6375294471505404
	14315	10630	0.6373637689636704
	15059	9762	0.637032516460696
	14788	10180	0.636853534541465
	14572	10301	0.6368165865472157
	22105	10277	0.6362856564162671
	13848	10053	0.636232545634973
	18120	10365	0.6360004789928929
	16055	10043	0.6359981156797139
	15117	10313	0.6359893168528029
	14145	10141	0.6359808506954112
	252114	31767	0.6355576556978296
	26215	11277	0.6353326807301314
	15323	10094	0.635000243922045
	13419	10373	0.6347930761631212
	35830	11147	0.6347731563020587
	15845	10546	0.6345010270524752
	17728	10061	0.6343152887357906
	14043	10224	0.6341616892875361
	18982	10361	0.6336301933116417
	14402	9702	0.6333294761247257
	27950	10714	0.6331734526264976
	16436	10209	0.6331526696281822
	14900	9673	0.633113450531128

LEMMA	FREQUENCY	RANGE	UDP
	14371	10171	0.6330208975965026
	15015	9921	0.6330172436216084
	14659	9992	0.6329762136568895
	13079	9984	0.632970885299484
	13402	10000	0.6329102726089664
	18144	10008	0.6328678991608488
	18170	10203	0.6327703065206036
	20467	10322	0.6324579177935619
	13112	9922	0.6323616321690009
	13094	9890	0.6320485045403406
	15224	9761	0.6319536942603103
	18357	9969	0.6318472859042107
	13058	9316	0.6310836172771022
	15936	10142	0.6310753406027976
	12856	9796	0.6309206679591
	14189	9964	0.6307658954717199
	14294	10242	0.6305407031119609
	17461	10079	0.6303642267030325
	13871	9724	0.6300794449888477
	13243	9667	0.6300372871659399
	14766	9914	0.6299231010632634
	15678	10164	0.6296962364801131
	27584	10350	0.6294577542793848
	12504	9531	0.6294190373721095
	16497	9866	0.6293985240147305
	14356	9692	0.6293963852252049
	14950	9634	0.6293123722729199
	18519	9777	0.6292853814220976
	32318	10829	0.6292684006255549
	25210	10450	0.6290176556280616
	17658	9785	0.6290015587240825
	15106	9441	0.6289462904388844

LEMMA	FREQUENCY	RANGE	UDP
	16207	9891	0.6289375763697199
	11877	9398	0.6288659047368299
	19498	10136	0.6287532454016026
	22710	10575	0.6286980445586567
	15140	9960	0.6286547418767876
	15935	9761	0.6285210688292258
	12802	9306	0.628340491357844
	14982	9386	0.6282947312985864
	24262	10087	0.6282915649969693
	14696	9411	0.6281780833446231
	13210	9530	0.6281450778569049
	22578	9766	0.6280505889113301
	15699	9702	0.6280228725515788
	24568	9865	0.6276081780759255
	27791	9829	0.6273454868978963
	12924	9468	0.627143748122212
	13080	9321	0.6267669239967131
	15025	9242	0.6267395572448191
	20206	9844	0.6266106798517285
	18887	9677	0.6265302388902527
	15573	9390	0.6264745691760816
	14296	9540	0.6263396616075203
	15509	9185	0.626243092725253
	13514	9280	0.6260333170126584
	52201	11674	0.6258631736471245
	14861	9585	0.6258606337010565
	13755	9557	0.6256709005921047
	19366	9858	0.6252465957131422
	14366	9531	0.6252128989927275
	13273	9438	0.6249713178626486
	13497	9576	0.6249458234012253
	16978	9232	0.6247997152286697

LEMMA	FREQUENCY	RANGE	UDP
	15526	9546	0.6247885971822662
	14289	9589	0.6246517235147988
	12043	9036	0.6246440976083485
	16418	9343	0.6245483334190125
	14769	9390	0.6244039584800193
	108693	17359	0.6242337750044071
	19027	9566	0.6240775861461408
	14132	9286	0.624076144534363
	14239	9071	0.6238463174447046
	13391	9172	0.6237391282139817
	15093	9252	0.623666206327153
	21671	9337	0.6233666638247364
	12844	9012	0.6231106401685387
	24470	10022	0.6229912610502499
	12768	9169	0.622887595853157
	13710	9107	0.6227707513738887
	19464	9118	0.622516224062368
	14472	9273	0.6224521241977641
	11632	8976	0.6223452962874663
	13755	8933	0.6223190967082686
	13188	8819	0.6221384594715977
	11984	9151	0.6220896285225851
	12111	8928	0.6215892099096972
	12880	9033	0.6215815402105227
	12244	9093	0.6214685059104785
	12826	8867	0.6214370417895815
	17386	9105	0.6213662061136842
	20913	9252	0.6210193267729724
	13143	9067	0.6207150098824594
	11528	8693	0.6206236099421755
	13720	8773	0.6206086391732284
	11977	8918	0.6201722638438825

LEMMA	FREQUENCY	RANGE	UDP
	15419	9143	0.6200478094974897
	12053	8877	0.620044703033531
	12512	8786	0.6199315468765172
	15280	8825	0.6197141211113539
	13544	8853	0.6195560104659852
	13379	8735	0.6194960177864491
	14209	8982	0.6193206225152128
	12503	8574	0.6193160177125396
	12824	8729	0.6192660881795813
	12282	8955	0.6192574299509408

Appendix 2: Scripts

Appendix 2.1: HebrewLemmaCount.py

```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

import re
import os
import gzip
from collections import defaultdict


############################################################
# ----------------- INITIALIZE VARIABLES ----------------- #
############################################################

# Define path for topmost directory to search. Make sure this points to
# the correct location of your corpus.
corpus_path = './OpenSubtitles2018_parsed_single'

# Initialize dictionaries
lemma_by_corpus_dict = {}
lemma_totals_dict = {}
token_count_dict = {}
lemma_DPs_dict = defaultdict(float)
lemma_UDPs_dict = defaultdict(float)

total_tokens_int = 0
table_list = []

# Set size of final list
list_size_int = 5000
```

```python


##############################################################
# -------------------- DEFINE FUNCTIONS -------------------- #
##############################################################


# Open XML file and read it.
def open_and_read(file_loc):
    with gzip.open(file_loc, 'rt', encoding='utf-8') as f:
        read_data = f.read()
    return read_data


# Search for lemma and add counts to "frequency{}".
def find_and_count(doc):
    corpus = str(f)[38:-4]
    match_pattern = re.findall(r'lemma="[ '"+[ -, doc)
    for word in match_pattern:
        if word[7:-1] in lemma_by_corpus_dict:
            count = lemma_by_corpus_dict[word[7:-1]].get(corpus, 0)
            lemma_by_corpus_dict[word[7:-1]][corpus] = count + 1
        else:
            lemma_by_corpus_dict[word[7:-1]] = {}
            lemma_by_corpus_dict[word[7:-1]][corpus] = 1


##############################################################
# --------------------- OPEN AND READ ---------------------- #
##############################################################

# Open and read all files. If calculating only for a specific language,
# comment out this code and uncomment the large block that follows.
#
```

```python
for dirName, subdirList, fileList in os.walk(corpus_path):
    if len(fileList) > 0:
        f = dirName + '/' + fileList[0]
        find_and_count(open_and_read(f))

############################################################
# ---------------- LANGUAGE-SPECIFIC BLOCK -----------------
#
# This large block of code is for creating a list using only movies #
# with a specific primary language (in this case, Hebrew). Be sure to #
# uncomment the relevant lines of code, and to comment out the block #
# above. #
#
#
# Create list of IDs for movies with Hebrew as primary language. #
# This makes use of a text file that must already exist with this list. #
#
# Hebrew_IDs_list = []
# with open('./Hebrew_originals.txt', 'r', encoding='utf-8') as f:
#     read_data = f.read()
#     Hebrew_IDs_list = re.findall(r'\s\stt[0-9]+\t', read_data)
# Hebrew_IDs_list = [line[4:-1] for line in Hebrew_IDs_list]
#
#
# Delete extra 0s at the beginning of Hebrew movie IDs. #
#
# for item in Hebrew_IDs_list:
#     if item[0] == '0':
#         Hebrew_IDs_list[Hebrew_IDs_list.index(item)] = item[1:]
# for item in Hebrew_IDs_list:
#     if item[0] == '0':
#         Hebrew_IDs_list[Hebrew_IDs_list.index(item)] = item[1:]
#
#
```

```
# Open and read files for movies with Hebrew as the primary language. #
#
# for dirName, subdirList, fileList in os.walk(corpus_path):
#     if len(fileList) > 0:
#         f = dirName + '/' + fileList[0]
#         folders = re.split('/', dirName)
#         if folders[len(folders)-1] in Hebrew_IDs_list:
#             find_and_count(open_and_read(f))
#
# ------------- END OF LANGUAGE-SPECIFIC BLOCK -------------
############################################################


############################################################
# --------------------- CALCULATIONS --------------------- #
############################################################

# Calculate token count per corpus
for lemma in lemma_by_corpus_dict:
    for corpus in lemma_by_corpus_dict[lemma]:
        token_count_dict[corpus] = token_count_dict.get(
            corpus, 0) + lemma_by_corpus_dict[lemma][corpus]

# Calculate total frequencies per lemma
for lemma in lemma_by_corpus_dict:
    lemma_totals_dict[lemma] = sum(lemma_by_corpus_dict[lemma].values())

# Calculate total token count
for corpus in token_count_dict:
    total_tokens_int = total_tokens_int + token_count_dict.get(corpus, 0)

# Calculate DPs
for lemma in lemma_by_corpus_dict.keys():
    for corpus in lemma_by_corpus_dict[lemma].keys():
        lemma_DPs_dict[lemma] = lemma_DPs_dict[lemma] + abs(
            (token_count_dict[corpus] /
             total_tokens_int) -
            (lemma_by_corpus_dict[lemma][corpus] /
             lemma_totals_dict[lemma]))
lemma_DPs_dict = {lemma: DP/2 for (lemma, DP) in lemma_DPs_dict.items()}

# Calculate UDPs
lemma_UDPs_dict = {lemma: 1-DP for (lemma, DP) in lemma_DPs_dict.items()}


##############################################################
# -------------- SORT LIST AND CREATE TABLE -------------- #
##############################################################

# Sort entries by UDP
UDP_sorted_list = [(k, lemma_UDPs_dict[k]) for k in sorted(
    lemma_UDPs_dict, key=lemma_UDPs_dict.__getitem__,
    reverse=True)]

# Create list of tuples with all values (Lemma, Frequency, Range, UDP)
for k, v in UDP_sorted_list[:list_size_int]:
    table_list.append((k, lemma_totals_dict[k], sum(
        1 for count in lemma_by_corpus_dict[k].values() if count > 0),
        v))

##############################################################
# ---------------- SORT-BY-FREQUENCY BLOCK -----------------
#
# Sort entries by raw frequency (total lemma count). To sort the final #
# list by frequency instead of UDP, comment out the above code within the #
# "SORT LIST AND CREATE TABLE" section, and also uncomment the relevant #
# lines of code in this block. #
#
```

```
#
# Sort entries by raw frequency #
#
# frequency_sorted_list = [(k, lemma_totals_dict[k]) for k in sorted(
#     lemma_totals_dict, key=lemma_totals_dict.__getitem__,
#     reverse=True)]
#
#
# Create list of tuples with all values (Lemma, Frequency, Range, UDP) #
#
# for k, v in frequency_sorted_list[:list_size_int]:
#     table_list.append((k, v, sum(
#         1 for count in lemma_by_corpus_dict[k].values() if count > 0),
#         lemma_UDPs_dict[k]))
#
# ------------- END OF SORT-BY-FREQUENCY BLOCK -------------
############################################################

# Calculate list size for 80% coverage and set that as the list size. Note
# that if the initial list_size_int (set near the beginning of the script)
# provides less than the desired coverage, it will default to that instead.
#
# added_freq_int = 0
# count = 0
# for k, v in UDP_sorted_list:
#     if added_freq_int / total_tokens_int < 0.8:
#         added_freq_int = added_freq_int + lemma_totals_dict[k]
#         count = count + 1
#     else:
#         break
# list_size_int = count

# Write final tallies to CSV file
result = open('./export/HebrewWordList2.csv', 'w')
```

```
result.write('LEMMA, FREQUENCY, RANGE, UDP\n')
for i in range(list_size_int):
    result.write(str(table_list[i][0]) + ', ' +
                 str(table_list[i][1]) + ', ' +
                 str(table_list[i][2]) + ', ' +
                 str(table_list[i][3]) + '\n')
result.close()

# Print final tallies. Uncomment this code to see the results
# printed instead of writing them to a file.
#
# for i in range(list_size_int):
#     print('Lemma: ' + table_list[i][0] +
#           '\tFrequency: ' + str(table_list[i][1]) +
#           '\tRange: ' + str(table_list[i][2]) +
#           '\tUDP: ' + str(table_list[i][3]))
```

Appendix 2.2: OMDb-fetch.py

```python
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

# import re
from sys import argv
import os
import glob
import omdb

# year = '1996'
script, year, id_start = argv

dirs = []
p = []


for name in glob.glob(
        '../OpenSubtitles2018_parsed/parsed/he/' + year + '/*/'):
    p.append(name)
# p = Path('../OpenSubtitles2018_parsed/parsed/he')
# p = list(p.glob('[198-199]*/*/*.xml'))

p = [os.path.basename(os.path.dirname(str(i))) for i in p]

for i in p:
    if i not in dirs:
        dirs.append(i)

for i in dirs:
    while len(i) < 7:
        dirs[dirs.index(i)] = '0' + i
        i = '0' + i
```

```

dirs.sort()

# for i in dirs:
#     print('tt' + i)

print('# ' + year + '\n' +
      'IMDb ID\tTitle\tYear\tLanguage(s)')


omdb.set_default('apikey', '906517b3')

for i in dirs:
    if id_start != '':
        if i > id_start:
            print('tt' + i + '\t', end="", flush=True)
            doc = omdb.imdbid('tt' + i)
            # if doc['language'] == 'Hebrew':
            print(doc['title'] + '\t' +
                  doc['year'] + '\t' +
                  doc['language'])
    else:
        print('tt' + i + '\t', end="", flush=True)
        doc = omdb.imdbid('tt' + i)
        # if doc['language'] == 'Hebrew':
        print(doc['title'] + '\t' +
              doc['year'] + '\t' +
              doc['language'])
```

Appendix 2.3: single_file_extract.py

```python
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

import shutil
import os

source = '../OpenSubtitles2018_parsed'
destination = './OpenSubtitles2018_parsed_single'

# Copy the directory tree into a new location
shutil.copytree(source, destination, ignore=shutil.ignore_patterns('*.*'))

# Copy the first file in each folder into the new tree
for dirName, subdirList, fileList in os.walk(source):
    for fname in fileList:
        if fname == '.DS_Store':
            fileList.remove(fname)
    if len(fileList) > 0:
        del fileList[1:]
        src = dirName + '/' + fileList[0]
        dst = destination + dirName[27:] + '/'
        shutil.copy2(src, dst)
```

Appendix 3: List of movies used

6 References

Vita

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam et turpis gravida, lacinia ante sit amet, sollicitudin erat. Aliquam efficitur vehicula leo sed condimentum. Phasellus lobortis eros vitae rutrum egestas. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec at urna imperdiet, vulputate orci eu, sollicitudin leo. Donec nec dui sagittis, malesuada erat eget, vulputate tellus. Nam ullamcorper efficitur iaculis. Mauris eu vehicula nibh. In lectus turpis, tempor at felis a, egestas fermentum massa.

Brezina, V., & Gablasova, D. (2015). Is there a core general vocabulary? Introducing the new general service list. *Applied Linguistics*, *36*(1), 1–22. https://doi.org/10.1093/applin/amt018

Brysbaert, M., & New, B. (2009). Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior Research Methods*, *41*(4), 977–990. https://doi.org/10.3758/BRM.41.4.977

Coxhead, A. (2000). A new academic word list. *TESOL Quarterly*, *34*(2), 213–238. https://doi.org/10.2307/3587951

Lison, P., & Tiedemann, J. (2016). OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, 7.

Nation, I. (2016). *Making and using word lists for language learning and testing.* Amsterdam: John Benjamins Publishing Company. https://doi.org/10.1075/z.208

Sorell, C. J. (2013). *A study of issues and techniques for creating core vocabulary lists for English as an international language* (Unpublished Dissertation). Victoria University of Wellington, Wellington, New Zealand.