Evaluating Utility of Vocabulary to Language Learners

Leonard Paál
Hochschule Karlsruhe
Supervising professor: Jannik Strötgen
Additional supervision by: John Blake (University of Aizu)

December 2024

Abstract

[Abstract]

Contents

Chapter 1

Introduction

ToDo: Write Intro

Put succinctly, we can pose this as a challenge of finding a computational method that can answer the question:

> *Which words provide the most language understanding in a given linguistic context?*

We can subdivide the overarching goal of word utility evaluation into three related, but not quite equivalent sub-goals:

- Evaluating the utility of a single word.
- Evaluating the utility of an (ordered) list of words to learn.
- Generating maximally useful (ordered) lists of words to learn.

The first and third goal are directly related, in that evaluating the utilities of words in a text trivially generates a useful list of vocabulary if we compile the list by aligning the words in order of their descending utility. Evaluating the utility of an ordered list of vocabulary is conceptually further removed from the other two sub-goals, but nevertheless, its implementation shares many aspects with them. For this reason, many chapters in this work contribute to all of these goals equally. Therefore, to avoid redundancy, we refer to the overarching goal of this work as "word utility evaluation" throughout the following chapters - which is meant to include these three sub-tasks.

1.1 Motivation

1.1.1 Role of Vocabulary in Language Acquisition

Learning a second language involves many different skills, often categorized into listening, reading, speaking, and writing. Another categorization may be vocabulary, grammatical skills, the ability to understand known words in various accents, understanding language when spoken at a fast speed. One skill that is required for any of these if the knowledge of vocabulary in the target language. A person with basic grammatical skills but no vocabulary has no ability to express themselves or

understand anything which they hear around them. On the other hand, a person familiar with rudimentary vocabulary but no grammatical knowledge may struggle with understanding complex sentences and sound unnatural when speaking, but can at least make sense of short phrases and express themselves. Thus, basic knowledge of vocabulary is clearly one of the most essential skills for using a language. This raises the question of which vocabulary should be learned first when starting out on the journey of language acquisition. This work attempts to contribute to answering this question.

ToDo: Mention various methods of finding useful vocabulary from viewpoint of learner: textbooks, apps, etc.

1.1.2 Context-specific Language Learning

ToDo: use this language report `https://www.babbel.com/press/en-us/releases/2016-01-20-Babbel_Global_User_Survey.html`

Learners of languages are typically interested in one or more specific aspects of the language. There is no such thing as an unbiased corpus than can fit every use case. Decisions must be made how much focus is given to everyday conversation, academic writing, writing pertaining to business like job applications etc. Many textbooks group vocabulary by topic, with a new topic being introduced with each lesson that student should ideally be able to converse in after completing the lesson. However, this way of introducing vocabulary has several shortcomings:

- Students spend time learning specific terms about the topic in one lesson while not learning even general vocabulary in other topics until much later.
- Knowing the words from previous lessons becomes a prerequisite for the more advanced material, especially because the terms from earlier lessons are used in example sentences for grammar. Thus, learners interested in learning the use of the language in one context will have a hard time of skipping earlier, less pertinent lessons to them.

(cut segment about frequency and TF-IDF here)

Thus, there is a need for further exploration as to how word utility can be calculated, using modern NLP methods involving artificial intelligence where necessary. Put more simply, this question may be reduced to:

What order or words, when learned, gives the learner the best set of words to understand and communicate in the language as quickly as possible?

1.1.3 Examples of Contexts and Words

Some examples for contexts that could be interesting to language learners include:

- Reading Wikipedia articles about a specific field (computer science, literature, biographies)
- Watching movies

- Travel to a country where the target language is spoken
- Doing business with a company from a country where the target language is spoken
- Cultural exploration (literature, religion)
- Finding friends from other countries

The different contexts for which learners might be motivated to learn a language differ in how easily corpora can be obtained about to mine patterns from. Movie subtitles and Wikipedia articles are easily obtained from sites such as opensubtitles.org and wikipedia.org. The words that might be relevant for travel are not as easily obtained: One might imagine an ideal scenario to collect data, in which a statistically relevant group of people travelling to the destination to be examined are randomly selected and equipped with microphones and cameras before the travel. During travel, one could record their conversations, conversations with people around them, and materials they attempt to read to navigate their journey such as train schedules, descriptions of tours, restaurant menus, street signs, etc. Lacking the funds to conduct an experiment for every possible language, this work is interested in finding a methodology to obtain data from readily available corpora and websites online that extracts relevant vocabulary from the source texts. Depending on the context, we can think about which of the following English words might be likely to appear frequently in the texts:

- Convert
- Cash
- Hug
- Dammit
- Y'all
- From
- Nineteen eighty-four
- Married

To examine a few examples: Words like "convert" occur frequently when looking at Wikipedia articles [1]. The word "cash" is likely to be useful for travelers, but in most other contexts, it would not be as relevant. "Y'all" is almost never used in formal writing but used abundantly in everyday speech in the southern United Stated of America and South Africa. "From", meanwhile, will be likely to be one of the most frequently used words regardless of context.

1.2 Challenges and Contributions

In this work, we make the following contributions:

[1] See "Wikipedia" corpus 2016, drawn from one million lines on `https://wortschatz.uni-leipzig.de/en/download/English`

- A workable definition of word utility (Section 2.1.2).
- The development of a framework for evaluating and generating vocabulary lists, by evaluating the utility of words to understand the meaning of a given text (Chapter 3).
- The analysis and selection of appropriate proxy tasks, corpora, and XAI methods, leading the implementation of this approach (Chapter 4), supporting an estimated 9 languages with an 40 languages without filtering of names from the lists.
- Analysis of the resulting vocabulary lists in comparison with each other, as well as frequency-based baseline methods (Chapter 5).

A major challenge in making computational approaches which generate maximally useful lists of vocabulary is that, to the best of our knowledge, no automatic process has yet been proposed which can measure how useful the words in a vocabulary list are. Thus, in order to evaluate the vocabulary lists produced by our generation approach, we develop our own evaluation method, which is based on similar ideas as our approach for list generation. However, we must also assess whether our evaluation approach itself is reliable, which is necessarily manual and subjective for lack of other evaluation measures.

A general theme of this work is multilinguality — we attempt to make our implementation applicable to the largest number of languages possible. For instance, we seek out sources of text to use as input to our data pipeline where many languages are available and labeled. This criterion limits the resources available. Another challenge in our particular word utility evaluation approach was finding a mechanism to automatically measure an AI model's understanding of texts, as this understanding is used to calculate a word's utility. For this, we let the AI model perform a task on the text, similar to a language exam in a school. However, finding tasks that require a broad language understanding to be performed but which can also be reliably evaluated automatically was a major difficulty.

1.3 Outline of Work

The following gives an overview of this work from a top-level perspective.

We begin by placing the work in context in Chapter 2, by first establishing how to the concept of "utility" is used in existing linguistic research about vocabulary list compilation, and explaining further basic concepts such as a *linguistic context*. With the use of these basic terms and concepts, the aim of this thesis is stated more precisely, followed by an introduction of the main fields of research involved in our solution to achieve this aim. Finally, we include a survey about concrete methods which are currently used to compile vocabulary lists, and how this work aims at improving them with the use of AI models.

Chapter 3 describes in detail our theoretical approach to both evaluating and generating vocabulary lists. To this end, we first transform our problem into a mathematical one. We then describe an empirical approach for evaluating how well a vocabulary list helps a learner to understand a given context, and how we

automate the approach by using Artificial Intelligence models to simulate a learner and corpora to simulate linguistic concexts. With the evaluation approach as a basis, we develop a second approach for generating vocabulary lists, using methods from the field of Explainable AI.

The technical implementation for both approaches is detailed in Chapter 4: After having established the principal components to the approaches in Chapter 3, we discuss the choice of these components, namely corpora, AI models, and Explainable AI methods: For each component, we first describe desiderata, and then make a number of concrete choices which were used in our experiments.

Chapter 5 discusses the evaluation of the vocabulary lists generated by our list generation approach. For this, we mainly utilize our own evaluation approach described in Chapter 3, but also make several subjective observations on the perceived utility of the lists from a human intuitive perspective. We evaluate how well the approaches perform on two different types of datasets, and how much time they require, using frequency-based approaches, which predominate the literature on vocabulary list compilation, as baselines. Additionally, we analyze how similar the lists are to one another, independent of their performance.

Finally, Chapter 6 outlines potential improvements to our approach to word utility evaluation, which could not be implemented within the scope of this work due to time constraints.

Chapter 2

Background

This chapter gives describes the theoretical background to estimating the utility of vocabulary. We first discuss how this work connects to research about vocabulary acquisition for second language learners in Section 2.1. After this, we define the exact aim of the work in Section 2.1.2. The following three sections explain the (partially overlapping) fields of Natural Language Processing, Machine Learning, and Explainable AI, tools from all of which are employed in our word utility evaluation approach laid out in Chapter 3. Finally, we examine current approaches for selecting vocabulary for language learners in Section 2.5, to show what methods are currently employed and how they could be improved.

2.1 Context of the Work

This section contextualizes the contents of this work by introducing theoretical considerations by linguistic researchers on the point of vocabulary lists. This serves to motivate our research, and makes clear what it does and does not aim to achieve, which is explicitly laid out in the second half of the section.

2.1.1 Linguistic Motivation

The issue of which words to learn when setting out to acquire a language is not a new one. Every textbook on language which does not focus exclusively on grammar must address the issue, and proactive learners will doubtless find themselves looking for ways to optimize the return on their invested studying time.

The Routledge Handbook of Second Language Acquisition [21] is a comprehensive reference work which addresses the various challenges in second language acquisition, including the role of feedback in language learning, the setting for learning and individual differences between learners. The chapter concerned with vocabulary acquisition gives three criteria for deciding the order in which words are taught to beginners:

1. Frequency
2. Usefulness

3. Easiness

The first criterion, frequency, is easy to justify: Learning words that appear with great frequency allows the learner to understand the maximum percentage of words in texts, and should thus be among the most relevant.

Secondly, there is the criterion of "*usefulness*". While it is acknowledged as a separate criterion from frequency, the authors do not define what constitutes usefulness, or how it might be measured.

The *easiness*, or *learnability* of a word is defined as how easy it is for a learner to acquire the word. The authors define it with respect to an individual learner since it is posited that the various linguistic backgrounds of learners influence which words they can acquire with ease. Apart from learner-independent factors influencing the learnability of a word such as word length, whether the learner already knows a cognate of the word will certainly also have an impact:

We can imagine a native German speaker attempting to learn English: The word "internationalization" may not be a particularly frequent word in most contexts and thus not seem too important to teach, but the German will likely recognize the word as a cognate of the German equivalent "Internationalisierung", and acquire the word with ease, giving a justification for teaching the word early if we wish to impart the most language knowledge as quickly as possible.

In their 2019 paper [10], He and Godfroid pick up the above three criteria in order to group words into clusters which should help determine their priority in vocabulary learning. However, again it is not defined what constitutes usefulness and no way to measure it is suggested beyond "human intuition". In fact, it is put in contrast with frequency, which may be derived from corpus data, whereas usefulness cannot.

It can be seen from the previous examples that a concept of *usefulness* exists in discussions about what the best order in which to teach vocabulary might be, but that it has hitherto not been examined empirically. The aim of this work is to find ways to evaluate word usefulness, or utility, automatically. Therefore, the following section defines utility and related concepts more carefully, in order to state the aim of this work more formally.

2.1.2 Statement of Goal

Generally speaking, this work addresses the problem of evaluating how useful it is to learn a given word in the target language of a language learner. This helps us in generating ordered lists of vocabulary to learn which aid the learner in gaining competency in their target language quickly. Since different language learners learn languages for different reasons [1], we attempt to take into account the learner's motivation to tailor vocabulary specifically to the individual. ToDo: Might have to remove this footnote depending on Motivation chapter This stands in contrast to traditional learning resources such as textbooks, which, by their linear and static structure, cannot take individual differences in interests of learners into account. To

[1]Statistics on motivations of language learners: `https://www.babbel.com/press/en-us/releases/2016-01-20-Babbel_Global_User_Survey.html`

make this goal more specific, we define several concepts that help us speak more precisely about this goal and how it may be accomplished. These concepts are used throughout this work.

Linguistic Context

The ways in which people interact with language (a "context") can be described from many different perspectives and along different dimensions, such as the level of formality, the topic or the setting of the interaction. In this work, a *linguistic context* or just *context* refers to one particular such way of linguistic interaction. Examples of language contexts by topic are: News, football, crocheting. Examples of language contexts by situation are: Everyday conversation, scientific articles, watching movies. Contexts can be further subdivided into smaller contexts, or grouped across different lines: "News" could be subdivided into "international politics", "sports", "business", etc.

The concept of a language context is useful to a language learner since it enables them to prioritize learning vocabulary and grammar associated with the context they are interested in: Someone who is learning Arabic to better understand middle-eastern politics will have little use for words which are mostly associated with football.

The context can even determine the length and complexity of sentences that a learner must be able to understand: We will see in Section 4.3.6 that sentence in Wikipedia articles are about three times longer, on average, that those found in movie subtitles. Thus, learning for one or multiple specific contexts gives the learner a more concrete goal and thus better idea of what they must learn to achieve it.

In contemporary methods and tools for language learning, usually little emphasis is placed on learning for a specific context. This is partially due to the format of these tools: A static resource, such as a textbook, cannot cater to individual preferences of learners. Some publisher produce books on "Business English" or "Technical English", but each of these take manual effort to produce and can hence address only a few, relatively broad contexts, in addition to being scarce for less studied languages: For instance, despite being the 21st most spoken language in the world [2] , no textbook on Business Vietnamese is sold by any major US publisher such as Pears [3], HMH [4], or McGraw Hill [5]. Classroom lectures, where groups of students are taught in one course, can also only cater to individual preferences to a limited degree, especially if they are oriented around textbooks, for the above reason. In this work, we will present an approach for the compilation of context-specific vocabulary lists that requires minimal effort to expand to new languages or contexts.

[2] According to the Encyclopedia Britannica `https://www.britannica.com/topic/languages-by-total-number-of-speakers-2228881`, last accessed April 22, 2025

[3] `https://www.pearson.com`

[4] `https://www.hmhco.com`

[5] `https://www.mheducation.com`

Utility

In this work, the *utility* of a word is defined as the increase of language ability in a given context that a person gains by knowing a word versus not knowing it at all. Here, language ability refers to being able to understand more of texts they read in the given context, being able to speak about the subject, etc. It is easy to see that some words will be more useful than others given a context: In the context of international news, learning the word "war" will bring more understanding to the learner than "pear" or "polymorphism".

Proxy Task

Some of the concepts in the realm of language learning cannot be measured directly: This can be because they are psychological in nature ("understanding") or because they are too complex to be objectively measured by numbers ("language ability"). The above section defines word utility as an increase in "language ability", but to analyze this ability with computers, we must make be able to put a number on it, even though we lack both an agreed upon scale and unit of measurement.

To circumvent this issue, we use *proxy tasks*: In this paper, we use the term *proxy tasks* for tasks that test subjects (either human or AI models) can perform, where the test result (the *proxy measure*) is used to capture the abstract quantity under study. For example, as a proxy measure for a person's general spelling ability, we could make them choose from multiple spelling variants of a word, and use the accuracy with which they select the correct answers. This work employs AI models solving NLP tasks as an economic and repeatable substitute for real humans interacting with language, and uses the NLP tasks as proxy tasks to make "language understanding", and thus "utility", measurable concepts that can be computationally maximized.

Revised Statement of Aim

With the concepts above defined more precisely, we can state the goal more accurately, namely, creating computational methods which can find words that have the maximum *utility* given a particular *language context* by means of *proxy tasks*. With the words in cursive filled in, this translates to: Finding words that provide a language learner with the most language understanding in the specific situations they are interested in, requiring them to learn the least possible amount of words. Since the understanding of a learner cannot be directly measured, we use tasks that check the learner's understanding as an imperfect but necessary approximation. The following sections discuss the fields addressed in achieving this aim, and highlight relevant aspects from them that contribute in later chapters to conceptualizing and implementing a solution.

2.2 Natural Language Processing

While the aim of the work is to develop methods to automatically compile vocabulary lists that serve human learners, the technical implementation of this necessarily will process human language with the use of computers. This domain is called *Natural*

Language Processing, a field defined in The Handbook of Computational Linguistics and Natural Language Processing as the engineering domain of Computational Linguistics [1]. Natural Language Processing, often abbreviated as *NLP*, is both used in Computational Linguistics (the field concerned with analyzing natural language through language data) and as well as in practical applications such as Chatbots, Machine Translation, Speech Recognition etc. [14]. This work addresses both aspects, as it is an undertaking of Computational Linguistics through the analysis of how AI models perform typical **NLP tasks**.

2.2.1 Corpus

The typical way Computational Linguistics empirically analyzes language is through the use of **corpora**. A common definition of a corpus can be seen in [11] as "an electronically stored collection of samples of naturally occurring language", which "can be a test bed for hypotheses and can be used to add a quantitative dimension to many linguistic studies".

Recently, many large corpora have been compiled and are freely available to the public. Corpora differ from each other mostly in their source and method of compilation. Depending on their source, some corpora may serve as examples of language being used in a language context, such as news or movie subtitles. The idea of using context-specific corpora to compile context-specific vocabulary lists has been used in the past, especially to create lists of academic vocabulary for specific academic fields [41] [8]. In the implementation and evaluation of this work's approach to word utility estimation, we use several such context-specific corpora, as they present a way to analyze how language (and more specifically, vocabulary) is used in the linguistic context of the corpus.

2.2.2 NLP Task

This work attempts to make use of not only static data in the form of corpora, but also the interaction of AI models with those corpora in order to gain insights into word utility. This active part of Natural Language Processing consists of performing **NLP tasks**. Typical NLP tasks include [14]

- Sentiment Detection: Given a text, estimate the emotional state of the author.
- Masked Language Modeling: Given a text with a word blanked out, estimate what the word should be.
- Machine Translation: Given a text, translate it another language while preserving meaning as faithfully as possible.

Humans are not trained from childhood to do any one language "task": They can have conversations, answer questions about things they have seen, etc. In contrast, the interaction of computers is usually more rigidly defined by specific NLP tasks, and AI models are trained to perform one or several of these tasks.

For the purposes of this work, an *NLP task* is a function with a specific input and output format, where at least one of the two formats takes the form of natural language. We utilize NLP tasks as a way for AI models to interact with natural

language, enabling us to analyze how certain words influence this interaction; the exact approach is described in Chapter 3.

2.2.3 NLP Preprocessing: Tokenization

Language presents itself in continuous form in most situations: When listening to spoken language, it is not obvious where one word ends and another begins. Likewise, written texts we find online or in books are not necessarily subdivided into its semantic constituents. While words in the written English language are mostly separated by spaces, a writer may choose to create a new hyphenated word sequence on the spot as necessity demands.

Further complicating the issue of where to separate words is the fact that many non-European languages do not use spaces in their spelling (e.g. Japanese, Mandarin Chinese) or use spaces for a different purpose (separating syllables in Vietnamese, separating sentences in Thai). For these reasons, *tokenizers* are used in Natural Language Processing to divide continuous texts into individual words [14].

Splitting continuous text into distinct words serves following purposes in this work:

- Making statistics from word counts (e.g., counting which words occur many times).
- Assigning estimated utility to words.
- Being able to remove individual words in text inputs to AI models, in order to test what effect removing a particular word has on the output.

Speaking more precisely, tokenizer divide texts into *tokens*, which do not necessarily correspond to words [14]: For instance, the `bert-base-uncased` tokenizer splits the word *enmity* into the sub-word tokens *en* and *##mity* (sub-word tokens which are not the start of a word begin with *##* in their text representation for BERT tokenizers). As we are interested in counting complete words, we must therefore merge sub-word tokens back into complete words when they occur in our data pipeline. The details of this are discussed when presenting our implementation in Chapter 4. In order to provide utility evaluation for words in a wide range of languages, this work employs tokenizers which can handle language of different syntactic structures, and where the tokenizer outputs sub-word tokens, we usually merge the tokens into complete words.

2.2.4 NLP Preprocessing: Named Entity Recognition

In this work, we attempt to compile lists of vocabulary by processing corpora with various methods. However, these corpora contain some words which are typically not desirable in vocabulary lists, such as the names of people, organizations, and places. For this reason, we employ Named Entity Recognition [14] (*NER* for short) to handle such words in our approach, and prevent them from appearing in our generate vocabulary lists. A named entity can be defined "anything that can be referred to with a proper name" [14]. Consider the following sentence:

Angela Merkel, the chancellor of Germany, visited Karlsruhe last month.

This sentence contains the personal name *Angela Merkel*, the country name *Germany* and the city name *Karlsruhe*. Because personal names such as *Angela Merkel* are generally not seen as vocabulary to learn, when we compile a list of English vocabulary from a corpus containing the above sentences, we filter such names from the lists. The names of places can be useful vocabulary, such as in the case of *Germany*, because it differs between English and other languages (e.g., German: *Deutschland*, Spanish: *Alemania*). However, our tool for Named Entity Recognition cannot distinguish between the names of countries and other places such as *Karlsruhe*. To prevent the vocabulary lists from containing many names of cities, we therefore filter out all recognized names of places, including those of countries. This is similar to the filtering of named entities performed in the KELLY project [18]: In it, NER was employed to filter out proper names from its lists of vocabulary. The only named entities allowed were the names of countries, which were manually added back into the lists after the automatic filtering.

The domain of Natural Language Processing has in the past been dominated by rule-based methods for processing language. However, natural language is a very complex subject which is not easily captured by a priori rules [15]. For this reason, methods based on statistical reasoning on large amounts of linguistic data handle many problems better than manually fixed rules. Beyond statistical methods, still further advances have been made in the field of Natural Language Processing through the use of AI models such as Neural Networks [15]. In order to exploit the state-of-the-art tools from the realm of Natural Language Processing, this work makes extensive use of such AI models, and therefore the next section therefore briefly introduces the field of Artificial Intelligence.

2.3 Artificial Intelligence

Elaine Rich, in her 1983 work "Artificial Intelligence", defines Artificial Intelligence as "the study of how to make computers do things at which, at the moment, people are better" [32]. An introductory article by IBM describes it as "a technology that enables computers and machines to simulate human learning, comprehension, problem-solving, decision-making, creativity and autonomy"[6]. This is opposed to traditional algorithms which can only follow algorithms which have been explicitly programmed.

2.3.1 AI Models

The building block of Artificial Intelligence is called an **AI Model**, which is a model using training data to learn patterns, and then used on inputs that were not identically present in the training dataset. AI models are used in this work to simulate an agent interacting with language. The (pre-trained) AI model is thought of as a test subject in possession of linguistic skills that can be studied. Through its training process, AI models learn patterns that are generalizable across the problem domain. If these patterns resemble the knowledge that humans gain when they learn a new ability, it may be possible to analyze the interaction of AI models with

[6] `https://www.ibm.com/think/topics/artificial-intelligence`, last accessed February 26th, 2025

data so that a human learner can learn from their behavior, similar to how humans can observe experts in their domain and learn from their behavior. The field which provides these methods of analysis is Explainable AI, which is explained in Section 2.4.

The current de facto standard for recent AI models is the *Neural Network* architecture [14], which attempts to imitate the neural makeup of the human brain. Such models achieve state-of-the-art results on most NLP tasks [15], and all models used in this work are Neural Networks for this reason. However, within this architecture there exist further subtypes of AI models, the most important for this work is the **transformer model**.

2.3.2 Transformer Model

Since their invention in 2017 [38], transformer models have brought impressive gains in performance to many fields, including Natural Language Processing. They are a particular type of deep neural network characterized by a mechanism called *attention*. This chapter briefly introduces this mechanism and how it is used in this work as one way to assign utility to words.

Typically, a transformer model features an encoder and decoder, each of which consist of an attention layer and feed-forward (i.e, non-recursive) neural network. Figure 2.1 shows the transformer architecture (featuring both an encoder and decoder). Due to their lack of recursion seen in networks such as LSTMs and GRUs, transformers are highly parallelizable.

The encoder consists of a so-called *self-attention* layer and a feed-forward neural network. The self-attention layer assigns weights to relationships between tokens, which contain information about how much the tokens are related to one another. This information helps the transformers to "focus" on important tokens, or important relationships between tokens. For generative transformers, there is also a decoder, which takes as inputs the outputs from the encoder network. It then uses another attention layer (using *cross-attention*) to assign weights to the embeddings before feeding them into another feed-forward network.

Many transformers feature a mechanism called *multi-head* attention. This means that the relationships between tokens are given weights from multiple *heads*. These individual heads "learn to perform different tasks", the combination of which helps create a more nuanced understanding of the token relationships [38]. Attention is one possible way to attribute importance to inputs [20], and thus utility to words in our case. How exactly this proxy metric is calculated is discussed in Section 4.4.

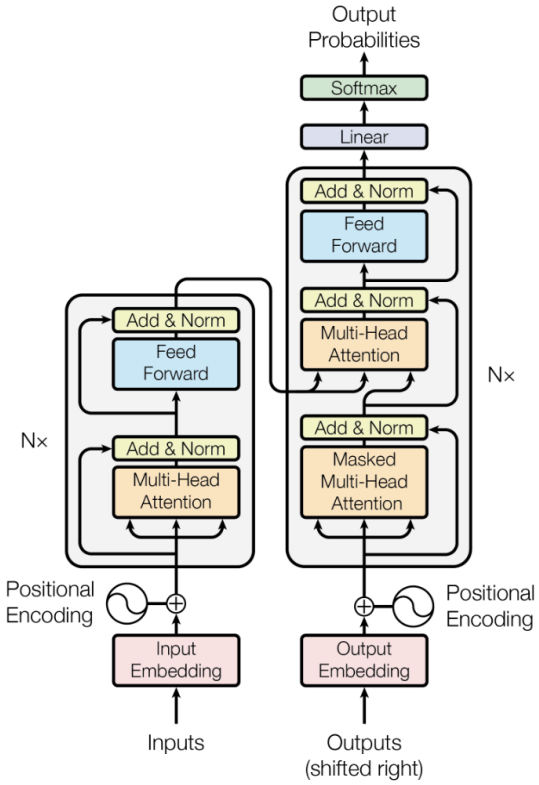

Figure 2.1: The Transformer Architecture. (Source: *Attention is All You Need* [38])

2.3.3 Summary

This section has introduced the important AI models used in this thesis, namely the Neural Network and one of its most effective subtypes, the transformer. While these types of AI model have driven many recent improvements in performing common NLP tasks, deep neural networks stop being readily understandable to humans rather quickly once the number of neurons and layers is increased. We are interested in how the inputs (words) influence the outputs of AI models when performing NLP tasks. This kind of analysis is the domain of **Explainable AI**, which is why the next section briefly introduces this last necessary field for the approach put forward in this work.

2.4 Explainable AI

Explainable AI (abbreviated: XAI) is the fields of research focused on making the decision-making progress of AI models more transparent and understandable to humans, to enable us to reason about the AI model's decisions [39]. This can be useful to check if the decision process contains social biases, or if it is based on wrong patterns learned from skewed training and test data (overfitting). By analyzing the decision process of high-performing AI models, this work attempts to extract useful information about how the language is processed by them which may be useful to humans as well. For this analysis, XAI is used as a tool to find out which inputs influence the performance of AI models the most. This section therefore first introduces important distinctions between XAI methods, and argues which types

should be used for our purpose. The result of this evaluation can be seen in Table 2.1.

2.4.1 Distinctions between XAI Methods

There are several categories of XAI methods that are relevant in this work, because they determine not only the purpose of the method, but also which XAI method can be used on which AI model. This section introduces some of the distinctions and argue why some types are more appropriate to the aim of this thesis than others.

As was mentioned before, this work is interested in analyzing the influence of words in AI model input on its output for the purpose of finding out which words might be the most useful for human learners as well. The question of which inputs are the most important for a model to reach its output is the domain of **feature importance explanations**, a sub-field of XAI concerned with "ranking the features by importance" [25]. Thus, all XAI methods in this work belong to this group.

Feature importance explanations can be further subdivided into two groups: **Feature selection** methods explain the model's decisions by trying to predict a subset of all input features which are deemed "important". **Feature attribution** methods attribute a concrete importance score $a_i \in \mathbb{R}$ to each input feature x_i. This work uses feature attribution methods exclusively, as the scores allow us to rank words in a list by their importance.

Another important characteristics of an XAI method is whether it is **model-agnostic** or **model-specific**. A model-agnostic XAI method can be used no matter what AI model is used, whereas a model-specific method is limited to being used to a specific type of AI model, such as transformers, decision trees, or neural networks. Model-agnostic models do not look inside the model (e.g., they do not examine weights in neural networks) but only perturb the inputs and observe changes in the model output. An advantage of model agnostic explanations is that they can be used on any AI model However, many recent state-of-the-art AI models are built using the transformer architecture, hence methods specific to transformer models can also be fairly broadly applied. A major upshot of model-specific explanations is their efficiency: Model-agnostic approaches require us to perturb the input to observe differences in the output, and thus to run the model multiple times for a single explanation, which can constitute a great computational effort. On the other hand, model-specific methods typically only require one run of the model, as they can look into the model itself for an explanation. For these reasons, both model-agnostic and model-specific approaches are used in this work.

Another feature of XAI methods is the scope of its explanations: **Local** XAI methods explain why a model output was generated for one particular input. A **global** method attempts to reason about the model's behavior in general, independent of any particular input data. This work is interested in observing the interaction of the AI model with a particular corpus, not only the AI model. For this reason, all methods used in this work are local methods. However, while local explanations only explain a single decision of the model (which, in the work, is the interaction of the model with a single sentence), we aggregate the individual decisions across the corpus to gain a bigger picture about this interaction.

The transformer attention mechanism introduced in Section 2.3.2 has been used as a model-specific XAI method [20] [19] , and is used in this work as one among several XAI methods employed to gauge the impact of words in the input to the output of (transformer) AI models. The use of attention as explanation, too, is discussed in more detail when describing the implementation of our approach in Section 4.4.

Table 2.1 summarizes the use of XAI in this work.

Attribute	**Type Selected for this work**	**Reason for selection**
Model-Specificity (Model-Agnostic or Model-Specific)	Both Model-Agnostic and Model-Specific	Model-Agnostic methods: Broad applicability; Model-specific methods: Computational efficiency
Importance Explanation (Feature attribution or Feature selection)	Feature Attribution	Use of scores to order words in vocabulary lists
Scope (Local or Global)	Local	Attain explanations taking into account the corpus

Table 2.1: XAI methods used in this work

2.5 Ranking Vocabulary Lists

As described in Section 2.1.1, the Routledge Handbook of Second Language Acquisition cites the concepts of frequency, usefulness and difficulty as widely accepted criteria to decide when to teach which words. To the best of our knowledge, all previous methods for making lists of vocabulary for language learners use similar criteria. They can be classified roughly into two categories based on which of the previously named criteria they employ: Approaches which use frequency have the benefit of being easy to implement on a computer and putting a simple number on each word.

Previous approaches which have aimed at generating lists by a utility concept outside of frequency have relied on human experts. This section introduces concrete methods from both categories, and examine each method's strong and weak points. It is this work's contention that AI models make it possible to mechanistically calculate utility in addition to frequency, and that this could be a possible point of attack for refining previously accepted approaches for vocabulary list compilation.

2.5.1 Expert Judgement: Reading Rockets Tier System

This section introduces a traditional approach to vocabulary list generation. While it does put forth criteria for vocabulary selection, they are not considered possible to

calculate mechanistically, and thus it relies on the judgement of individual language experts (teachers) to find useful words for their students.

It is seen in the recommendations for vocabulary selection by the initiative Reading Rockets: Reading Rockets [7] is an American national public media literacy initiative by WETA Washington, D.C. [8], a television station of the non-commercial public broadcaster Public Broadcasting Service [9].

As such, one of its aims is to help find reading materials appropriate for teaching children of school age words they are not yet familiar with but which are useful and important, an aim similar to that of finding useful vocabulary for second language learners. Reading Rockets has made public several recommendations for how to find useful vocabulary to teach children as well, with the intention of helping individual teachers select vocabulary for their students. For this purpose, Reading Rockets uses a tiered system to sort vocabulary items into three tiers: [10] Tier One items are words which most children know before even entering primary school and thus need not be taught. This includes words such as *happy* and *baby*.

Tier Three items are words which are useful mostly in one specific domain, such as *peninsula* or *lathe*. These are not high frequency words, and thus Reading Rockets conclude that these words are best learned not in isolation, but rather "when needed in a content area".

Tier Two words are fairly high frequency words that are not so common as to be known by all children, but still useful across many domains, such as *coincidence*, or *absurd*. For this reason, Reading Rockets deems that "instruction in these words can add productively to an individual's language ability." [10]

This system of triage can be useful and could be made to fit with the aim of teaching second-language learners too, by teaching Tier One words first, then moving on to Tier Two and Tier Three words. Because the system proposed by Reading Rockets is a manual one, it requires some effort on the side of a teacher who would like to apply the system to their classroom. In the next section, we discuss various vocabulary selection methods that utilize analysis of existing textual data to arrive at vocabulary lists.

2.5.2 Frequency-Based Methods

To the best of our knowledge, current algorithmic approaches for vocabulary list generation all rely on frequency as their main criterion for list ordering. However, the raw number of occurrences is not the only metric which falls into this category: The literature offers at least three frequency metrics for vocabulary list creation, namely *Raw Frequency*, *Relative Frequency* and *Average Reduced Frequency*. This section therefore introduces and discusses these approaches and their advantages and disadvantages. Furthermore, we discuss *TF-IDF*: A metric which, to our knowledge,

[7] `https://www.readingrockets.org/`

[8] `https://weta.org/`

[9] `https://www.pbs.org/`

[10] `https://www.readingrockets.org/topics/vocabulary/articles/choosing-words-teach`, last accessed on March 15, 2025.

have not been used to create vocabulary lists, but for a similar purpose, such as finding words in a document that most characterize its contents.

These computational methods generate vocabulary lists by using corpora and computer-aided language processing (mostly tokenization) to compile vocabulary lists. Compared to the expert-based approach explained in the previous section, they are closer related to the methods that this work strives for, and are used as points of comparison when evaluating our own approaches.

Raw Frequency

The raw frequency of a word is simply the number of times it occurs in a given text. The idea of making lists of vocabulary by ordering words by their number of occurrences in a corpus is not new: In 1953, English language teacher Michael West published his "General Service List of English Words" [24], and the frequency of the words in a corpus was one of the criteria used to determine its members and ordering.

Nation and Waring are prominent proponents of frequency-based approaches: In their 1997 paper "Vocabulary size, text coverage and word lists" [26], they affirm that the frequency with which a word appears in the target language should be used a metric for its importance to the learner, or utility, and thus teaching high-frequency words should be the focus when teaching beginners, claiming that: "Frequency information provides a rational basis for making sure that learners get the best return for their vocabulary learning effort."

Herein we can see the appeal of frequency as a basis for vocabulary selection: It gives the reader the highest *coverage* of texts, i.e., the percentage of words in the texts they read. However, focusing on maximum-frequency words to achieve text coverage does not take into account that the most frequent words in the language are typically words with little meaning by themselves: The following words are the most frequent ten words on Wikipedia according to a count by Leipzig University [11]: "the", "of", "and", "in", "to", "a", "is", "was", "the", "for".

Herein, we can see can issue with using frequency as the exclusive criterion for vocabulary selection: We do not contend that the above words are useful and among the most important words to know for an English learner. However, if we were to teach these as the first words, the learner would not be able to understand the meaning any sentence, nor make a complete sentence of their own design. This is because they are prepositions, articles, and copula verbs, which by themselves do not allow for the construction of grammatically correct sentences, nor convey much meaning. Such words are called *stopwords* in the realm of Natural Language Processing, and they are often filtered out from the inputs for AI models since they are not necessary to perform many NLP tasks [14]. Thus, knowing only these words is not sufficient for comprehending most texts.

[11] See "Wikipedia" corpus 2016, `https://wortschatz.uni-leipzig.de/en/download/English`

Average Reduced Frequency

The raw frequency count can give indications about which word in a long corpus are important to know. However, consider two words that both appear in a corpus n times, but one of them only appears in one small section, while the other is distributed evenly across the corpus. In such a case, we might draw the conclusion that the former is less generally useful than the latter.

This is the thought behind **Average Reduced Frequency** (ARF) [33]: If a word occurs n times throughout the corpus, and these occurrences are evenly distributed throughout the corpus, its ARF is roughly n. However, the more localized its occurrences are, the more its ARF approaches 1. The exact formula can be seen in equation 2.1.

$$ARF = \frac{1}{v} \sum_{i=1}^{f} \min\{d_i, v\} \tag{2.1}$$

where:

f is the raw frequency of the word,

d_i is the distance between each occurrence of the word (included the cyclic interval between the first and last occurence in the corpus), and

v is the average number of tokens between occurrences.

Average Reduced Frequency has been used for the purpose of compiling generally useful vocabulary lists: The KELLY project was a project by multiple European universities aimed at "developing language learning word cards with a language's most frequent words corresponding to the Common European Framework (CEFR)" [12]. KELLY employed ARF along with the raw frequency and relative frequency of a word in corpora to find useful words and thus generate lists of vocabulary [18].

TF-IDF

The previous sections have introduced proxy metrics for word utility based on how often a word appears in a (single) large corpus. To the best of our knowledge, all current algorithmic methods used for vocabulary list compilation are of this type. The issue with these approaches is that they tend to rate stop words the highest, as these are very high frequency words in any language. Because we are concerned with context-specific language learning, we could attempt to normalize the frequency of a word in a context-specific corpus with its frequency in a more generic background corpus. This is the idea of the **TF-IDF** (**T**erm **F**requency - **I**nverse **D**ocument **F**requency) [28] metric:

It is calculated on a document level: Given a collection of documents, we calculate the TF-IDF of a word in a document by taking the number of occurrences of the word in the document (Term Frequency), but discounting words that appear in a large number of documents overall (multiplying by the Inverse Document Frequency). Specifically, we used the following standard formula (taken from [28]):

[12] `https://spraakbanken.gu.se/en/projects/kelly`

Given a term t, a document d, a collection containing N documents, we define $\text{tf}_{t,d}$ as the number of occurrences of t in d, and df_t as the number of documents in which t appears at least once. Then our TF-IDF given by:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \log \frac{N}{\text{df}_{t,d}} \tag{2.2}$$

The TF-IDF metric is usually employed for finding keywords in a document, i.e., a small number of words that characterize its contents [28]. TF-IDF can be thus be seen as a proxy metric for how important a word is for the overall meaning of the document. We can see that there is a potentially paradoxical relationship between the raw frequency and TF-IDF, as generally frequent words achieve a high rank for the raw frequency metric, but a low rank for TF-IDF, leaving us to wonder which of them (if any) is a good proxy metrics for word utility.

Another potential downside of TF-IDF in comparison with the manual approach seen in Section 2.5.1, is that, while it can identify words that characterize the text, it does not take into account the semantic relationships between the words in a text. Thus, if a language learner studies words in order of their TF-IDF in a document they are interested in, the words may well aid in identifying the topic the document, but may not be as useful at finding out what the message conveyed about the topic is. For example, "not" is a highly frequent word in English and thus will have a high IDF and low TF-IDF score in most documents. But it is essential to know, as it can completely invert the meaning of a sentence. Our approach to word utility evaluation is an attempt to overcome this shortcoming of pure word counting, by taking into account additional semantic information from texts which can be extracted using recent AI-based methods.

2.5.3 Summary

Current methods for compiling lists of vocabulary do not exploit recent developments in AI technology and thus suffer from several shortcomings: Automatic methods are exclusively based on counting words, without taking into account their meaning or relationship between each other. Furthermore, they are usually used on very large corpora for vocabulary list compilation. While their use on smaller corpora is possible, the long tail of the distribution means that many words will have frequencies of 2 or 1. These words cannot put in a more useful order with raw frequency counts.

Manual methods, such as the *Reading Rockets* approach, rely on the expert judgement of individual teachers, making them subjective and challenging to implement on a large number of vocabulary terms. In the following section, we will present a novel approach for compiling and evaluating lists of vocabulary, which attempts to surpass purely count-based methods with the use of AI models and Explainable AI.

Chapter 3

Approach

This chapter describes in detail our approach for word utility evaluation to fulfill the goal stated in Section 2.1.2. The problem is first put formally by putting the terms of word utility and vocabulary list efficiency into mathematical objects in Section 3.1. We then describe a hypothetical experiment that could be performed to evaluate word utility using human feedback in Section 3.2. After pointing out the unfeasibility of such a testing method, we suggest ways to perform the experiment with AI models instead of humans as the test subject in Section 3.3, using the technological building blocks referred to previously in Chapter 2.

With a method for utility evaluation established, we then point out why it is in itself not yet sufficient for actually generating useful lists of vocabulary that a human learner could utilize in Section 3.4. To solve this final problem, 3.5 puts forward a general approach for list generation.

3.1 Formal Problem Statement

To repeat the core statement from Section 2.1.2: The aim of this work is to create computational approaches to find words that have the maximum *utility* given a particular *language context* by means of *proxy tasks*.

Keeping in mind that this is done in order to sort words into vocabulary lists that can be used by language learners, we can conclude that the output of a proposed solution to this problem would be an ordered list of words. If this word list is ordered by descending word utility, we might speak of a *maximally efficient* vocabulary list. We could also put a number on *any* list of words which measures how efficient a vocabulary list is to gain understanding in a given linguistic context. A list should be evaluated as more efficient if useful words appear at the top of the list, and less efficient if less useful words are at the top, since such an order implies that someone learning the words in that order would not gain understanding at the highest possible rate.

We can subdivide the overarching goal of word utility evaluation into three related, but not quite equivalent sub-goals:

- Evaluating the utility of a single word.

- Evaluating the efficiency of a list of words to learn.
- Generating maximally efficient lists of words to learn.

This section puts utility and efficiency into mathematical terms, to provide a clearer explanation of the variables involved in these tasks and their relationship.

Proxy Task

Since utility is defined in terms of language ability, we first need a way to put a number on the language ability of a test subject. This is done by means of a proxy task: We can imagine a human test subject taking a language exam as a proxy task to find out their language ability:

$$\begin{aligned} t : \text{Human} &\to [0, 1] \\ s &\mapsto p \end{aligned}$$

where s is the test subject and t is the proxy task, with a possible performance score p between zero and one.

We can imagine many variables going into this function, such as the time when the test is taken (hopefully the human's performance would increase over time). However, this thesis addresses vocabulary learning, and so the vocabulary of the test subject is provided as an additional parameter to the function t to make a slightly modified function, t':

Proxy Task with Vocabulary

$$\begin{aligned} t' : \text{Human}, 2^W &\to [0, 1] \\ (s, V) &\mapsto p \end{aligned}$$

where W is the set of all words in the language, and $V \in W$ is the vocabulary of the test subject.

Vocabulary List Efficiency

We now have a function t' from the vocabulary to the score in the proxy task (the proxy metric for language ability). Using this function, we can define a measure for the efficiency of an ordered list of vocabulary l containing elements from W. A list of vocabulary is simply an ordered list whose elements are words of the target language, with no duplicates:

$$l := (l_1, l_2, \ldots, l_n) \quad \text{where } l_i \in W \text{ and } \forall i, j : i \neq j \implies l_i \neq l_j.$$

Note that by this definition, the list does not have to contain all words of the language.

If the test subject learns the word exclusively in order of the list l (and retains them perfectly), their vocabulary at any point will be that list up until some index k:

$$V_{l,k} := \{l_i \mid i \leq k\}$$

This thesis aims at finding vocabulary lists that improve the ability of a language learner at the fastest possible rate. We can call this property the efficiency of the list. To measure it, we can take a number k of words that a learner memorizes from the top of the list. An efficient list means that with a small number of words, the learner should achieve a high score on the language exam (see Figure 3.1). Therefore, we can define the k-word-efficiency of a list as the score that it allows the learner to achieve by learning its first k words.

$$e_{t,s,k}(l) = t'(s, V_{l,k}) \tag{3.1}$$

And with this definition of efficiency, we can define the condition for an optimal vocabulary list given a particular number of words:

$$l_{opt,t'}(s,k) = \arg\max_{l} \quad e_{t',s,k}(l) \tag{3.2}$$

Finding vocabulary lists that approximate an optimal list according to formula 3.2 is the aim of the rest of this work.

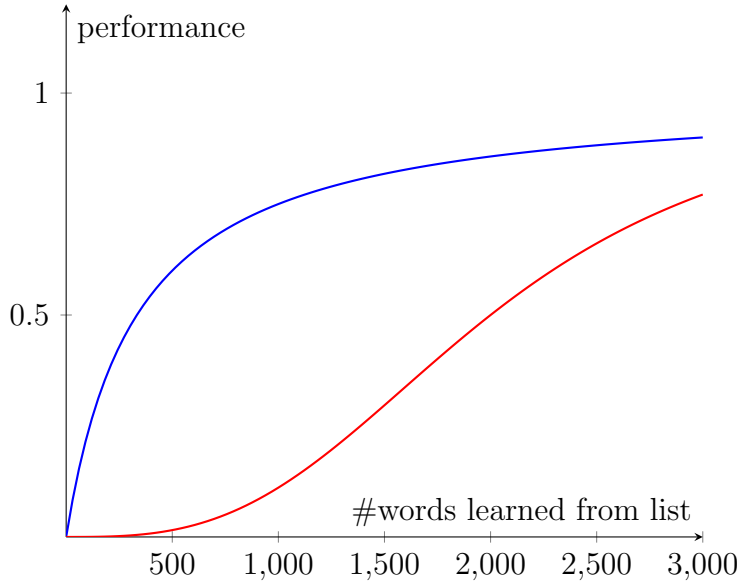

Figure 3.1: Visualization of vocabulary list efficiency: **Blue** represents the score with learning with an efficient list, **Red** represents the score learning from a less efficient list (i.e, less useful words at the top of the list)

3.2 Experimental Setup: Measuring Word Utility as Ability Improvement in Humans

With the formula 3.2, we now have set our goal more concretely. We are left with the question of how to design an actual experiment to test vocabulary list efficiency, and

also how to generate one that is approximates an optimal list. Hence, this section lays forth a hypothetical experiment design by using a human test subject to test efficiency, which will, however, be found to suffer from problems of feasibility, our solution to which is explained in Section 3.3.

Experimental Setup

Let us first consider an example scenario, where we wish to measure the efficiency of one vocabulary list l_A at $k = 200$, $k = 500$ and $k = 2000$ words: First, we would need to find a human test subject s who does not know any words of the target language at the outset. We then make the subject learn the words from l_A one by one. Once the subject seems to know 200 words, we perform a language exam and note their score. Finally, we repeat the process for 500 and 2000 words.

Problems of the Setup

While this test setup might take a long time to complete for high k values, it is still feasible. But if we wish to compare the efficiency of l_A with a second list l_B, we must measure how the same subject performs if they learn from the second vocabulary list, which presents is an unavoidable issue: **The test subject cannot arbitrarily forget the words they have learned from the first list l_A.** The experiment is thus not repeatable.

If we tweak the experiment such that each vocabulary list is learned by a different test subject, we introduce differences in capabilities between test subjects, making the evaluation of the list's efficiencies much more complicated to measure. To make up for these inconsistencies, we might boost the number of test subjects to the point where each list is learned by a statistically significant group of subjects, but this would cause a sharp increase in the cost of the experiment.

However, this theoretical test setup could be feasible using a non-human test subject: Using AI models and Explainable AI to analyze their interaction with language, we could not only evaluate, but even compile lists with drastically reduced costs. This approach is laid out in the following sections.

3.3 Experimental Setup with AI Model

This work is interested in finding vocabulary lists that language learners can use to gain linguistic competence in their chosen field in the least possible time. Section 3.1 has specified how with the help of a proxy task, we can define this efficiency. Section 3.2 has suggested a hypothetical experiment for testing the efficiency of a vocabulary list with human test subject. However, we have established that the experiment is not easy to set up consistently, as it cannot be repeated with the same test subject on different lists, and a large pool of test subjects either means lower comparability of the scores or significantly increased costs.

To circumvent this issue, this work proposes the following idea: To replace the human test subject in the experiment described in Section 3.2 with an AI model, facilitating a consistent and economical setup. The following paragraphs discuss

how, in such a setup, we can model the concepts from Section 2.1.2 such as linguistic context and utility with technological tools.

AI as a Test Subject

In recent years, AI models such as BERT have become highly adept at language-related tasks such as masked language modeling, question answering and sentiment detection [17]. Such models possess an "understanding" of language that extends to the meaning of words: They can recognize that two words such as "building" and "apartment" are close to each other in meaning, even though the words are dissimilar on a character level (this work makes no claims as to whether this ability stems from conscious understanding in the sense of the Chinese Room thought experiment [34], only that the models behaves as though they possessed such an understanding). Thanks to this semantic language "understanding", AI models surpass the purely statistical approaches that were the paradigm of NLP until recently [15]. Thus, instead of a human who performs a language exam as described above to analyze their increases in linguistic understanding, we could let an AI model perform an NLP task on a corpus instead.

If we run the model on a specific corpus and mask some of the words in the input, it is expected that the performance will decrease in comparison to the full input. However, presumably some words will have a larger impact on the performance than others. If we view removing words from the input as the equivalent of a human not knowing a word in a text, this performance differential can be a proxy metric for word utility and for human language ability.

Modified Experimental Setup with AI

With this new proxy metric for word utility in mind, we can propose a new experiment. To test the efficiency of a vocabulary list, we modify the setup from 3.2 as follows:

Let a pre-trained AI model perform its NLP task on a corpus. At the start, we mask all the tokens in the input to simulate a language learner who is a complete beginner and thus knows no words in their target language. We then progressively unmask the words from the vocabulary list in the input, and each time run the AI on the corpus to check the updated (and presumably improved) performance.

This yields a plot of performance over the number of unmasked words which shows how quickly the performance improves with unmasked words, similar to Figure 3.1. A quickly increasing performance corresponds to an efficient vocabulary list, whereas an inefficient list creates a plot with low initial gains in performance.

The modified experiment addresses the two issues proposed with the original setup: It is cheap when compared to using a human test subject and unlike the human, the AI model has no memory of previous tests, meaning we can run the experiment consistently every time. Next, let us see how the components in this setup interact with each other to model the concepts introduces in Section 2.1.2:

Influence of Components on Result

The evaluation of the vocabulary list efficiency in the experiment will depend on the following components:

- The AI model employed
- The input corpus
- The NLP task used

We have already addressed the role of the AI model in Section 3.3. The corpus and the NLP task performed by the model will also influence how quickly the AI model's performance increases with a given vocabulary list:

For example, when performing sentiment detection, words relating to emotion such as "hate", "like", "amazing" will matter more than for an Information Retrieval task such as Temporal Tagging. The corpus has a more obvious influence on the performance, since corpora differ in many aspects such as formality, topic, and format of the text, all of which are components of a linguistic context. Thus, by using diverse corpora, we can model various language contexts, and test word utility in those contexts. Once we have efficient vocabulary lists across those contexts, a language learners could choose one or several of these lists, based on what linguistic context is closest to their language learning goals.

Table 3.1 summarizes the correspondences between the concept and the technical component.

Abstract concept	**Technical Implementation**
Test subject	AI model
Language ability	Performance in NLP task
Language context	Corpus
Word utility	Impact of word on task performance

Table 3.1: Correspondence of abstract concepts to parts of implementation.

The next section discusses how we can not only test list efficiency, but make lists that approach an optimally efficient list.

3.4 Evaluating vs. Making Lists of Useful Vocabulary

The previous section describes an approach for evaluating the efficiency of vocabulary lists, which measures how quickly the list lets a learner improve their understanding in a specific context. While it can be seen as an improvement on the human setup because of its improved consistency and economy, it describes only the process of evaluation, not generation. Actually finding optimal vocabulary lists is theoretically possible by testing every possible vocabulary list with all words in the language. In practice, however, this would require an enormous amount of computational resources, as can be seen on a simple example:

Let us suppose we wish to find the most efficient vocabulary list of 1,000 words for some context. To speed up list generation, we can make a preselection by allowing only the most frequent 20,000 words in the context as candidate members of this list. This would yield a total number of $P(20000, 1000) = \frac{20000!}{19000!}$ possible vocabulary lists, all of which would need to be tested for efficiency to find the most efficient one.

We can optimize the evaluation process, however, since these tests consist of running the proxy task with a gradually increasing vocabulary, and each vocabulary is used in the evaluation for many vocabulary lists. For our example, a vocabulary is a set of words formed by taking the first n elements of the vocabulary list where $n \in [0, 1000]$. If we only consider the number of possible vocabularies with a cardinality between 0 and 1,000, we get $\sum_{k=0}^{1000} \binom{20000}{k} \approx \frac{1}{2} \times 2^{20000} = 2^{19999}$ runs of the proxy task. Clearly, this is still an unfeasible amount of computation.

It can thus be seen that finding optimal vocabulary list is much too expensive when performed with a brute-force approach. For this reason, the next section introduces an alternative method for finding efficient lists of vocabulary, based on XAI as a tool which extracts information about the linguistic skills of AI models.

3.5 Generating Efficient Lists of Vocabulary

The previous sections have shown the challenges in evaluating the efficiency of vocabulary lists with human test subjects, and suggested a repeatable approach utilizing AI models, NLP tasks and corpora to overcome these challenges. We have seen, however, that this approach does not suffice to generate efficient vocabulary. Therefore, to achieve cost-efficient list generation, this section advocates for adding a final component to our experimental setup, namely Explainable AI.

Explainable AI as a Tool of Analysis

The guiding question of this thesis is: *Which words provide the most language understanding in a given linguistic context?*. Section 3.3 has put forward an experimental setup in which the concepts of language understanding and context are simulated with technological components. By substituting these terms, the question is turned into: *Which words provide the highest improvements in performance of an AI model performing an NLP task on a given corpus?*.

Put like this, the question is very close to the questions that the field of Explainable AI seeks to answer. A typical question of Explainable AI would be: *Why, for a given input, does the model arrive at its output?*. To be more precise, this is a question for a *local* explanation (see Section 4.4). As mentioned in Section 4.4, we use feature importance attribution methods, which, for an NLP task, transforms the question into: *Which words in the input have the most influence on the model arriving at its output?*

By answering this question on a statistically relevant sample of individual inputs from a corpus, we can find out which words are the most essential for the AI model to perform its task in the corpus overall. If the words found in this way are close to those that would provide the most utility to a human learner as well, they could be used to create very efficient vocabulary lists as well.

Once we have calculated the utilities of words, we only need to sort the words by their utility to arrive at a vocabulary list that should be close to optimally efficient. It must be noted that this is only an approximation of a maximally efficient list. This is because, theoretically, the utility of two words combined could be higher than the sum of their individual utilities. In other words, our approach ignores possible synergistic effects between words.

By adding XAI as a tool for analyzing the interaction of the AI model with the corpus, we finally have an experimental setup for generating efficient vocabulary lists. This is not a replacement of the approach described in Section 3.3. Rather, this work uses the generation approach to make lists (Chapter 4), and the evaluation approach for testing which of these lists are the most efficient (Chapter 5). In the next section, we discuss the implementation of this approach. We present various choices for these components, and argue for some to be used over others, with the aim of making the utility estimated by the framework align as much as possible with utility to human learners.

Chapter 4

Implementation

Chapter 3 has formalized the problem and put forward a novel framework for finding useful words, consisting of two major functionalities: Vocabulary list generation, and vocabulary list evaluation. The list evaluation approach, described in Section 3.3, utilizes the performance of an AI model on an NLP task with a context-specific corpus as input as a proxy metric for language ability, in order to estimate how efficiently the vocabulary list may help a human language learner acquire competency. The list generation approach, proposed in Section 3.5, additionally uses Explainable AI as a tool for analyzing the interaction of the AI model with the corpus to compile vocabulary lists that approach maximal efficiency.

In this chapter, we describe our implementation, especially of the list generation approach. This is because the list evaluation method is used in Chapter 5 as one among several metrics for evaluating the efficiency of vocabulary lists generated by the implementation in this chapter. However, the evaluation with our approach uses the same AI models and Corpora as the list generation approach.

We first discuss the implementation of the list generation approach from a system design perspective in Section4.1, describing how we generate lists of vocabulary, without elaborating on the interdependencies that arise between concrete NLP tasks, corpora, and XAI methods. The choice of those three components in the system is then argued in the following sections, each of which features a section where we discuss desiderata of the specific component on the basis of our aim as stated in Section 3.1, followed by the selection of concrete components. Finally, we return to the holistic perspective in Section 4.5, where we present the complete implementation with all individual components integrated.

4.1 Data Pipeline

This section gives a top-level overview of our implementation for the list generation approach described in Chapter 5. As mentioned before, the main components of this approach are:

- An AI model, used as a test subject.
- An NLP task, the score in which is used as a proxy metric for the language

ability of the test subject.

- A context-specific corpus, to model a language context.
- An XAI method, which analyzes the interaction of the AI model with the corpus to determine word utilities.
- If the XAI method allows: A tokenizer, determining the words to be analyzed.

Thus, the implementation of the approach mainly consists of determining how exactly these components interact, as well as deciding which model, XAI method, and corpus, is used. The interaction of these components can be seen in pseudocode in Algorithm 1.

Algorithm 1 Efficient List Generation.

Require: corpus, model, xai_method
1: Initialize *line_word_utilities* with empty list
2: **for** each *line* in *corpus* **do**
3: $word\_utilities\_for\_this\_line \leftarrow$ xai_method$(model, line)$
4: Append $word\_utilities\_for\_this\_line$ to *line_word_utilities*
5: **end for**
6: *corpus_word_utilities* $\leftarrow$ aggregate(*line_word_utilities*)
7: *voc_list* $\leftarrow$ words ordered by *corpus_word_utilities*
8: **return** *voc_list*

The XAI method outputs the importance attributions in the form of one number for each unique word in each input line. The results are aggregated by taking the sum across all lines in the corpus, resulting in a list of word-utility pairs, which reflect the estimated utility of the word with respect to the entire corpus. To make a vocabulary list from these words, we simply order the words by their estimated corpus utility in descending order.

With this basic structure established, the following sections address the selection of individual components. Due to the interdependencies between certain components, not all are presented in separate sections. The NLP task performed depends on the AI model that we use, as most AI models are trained on only one task. Therefore, the choice of AI model is discussed together with the NLP task in the next section.

4.2 NLP Tasks and AI Models

Our approaches for list efficiency evaluation and for efficient list generation use an AI model with an NLP task to simulate a test subject performing a language exam. Because of this, the AI model and task chosen are a crucial component of these approaches, and their selection is among the most important in the implementation. The NLP tasks we choose also influence what corpora we can use for their execution: The task of next sentence prediction requires contiguous texts as input data, necessitating input corpora containing whole documents, not just individual sentences.

This section first puts forward criteria for selecting NLP tasks in Section 4.2.1 for word utility evaluation. We then discuss potential candidate NLP tasks, and discuss how appropriate they are for our approach. In the final selection, we decide on two tasks which, in the estimation of the author of this work, are suitable for our word utility evaluation approach, namely *next sentence prediction* and *sentence embedding*. These NLP tasks we choose dictate the format of our input data, i.e., the corpora we can use. Therefore, the selection of corpora is discussed after the choice of NLP is finalized.

4.2.1 Desiderata

This section puts forward four criteria for selecting NLP tasks for word utility evaluation, arising from our underlying goal of using the task as a proxy for real language interaction of a human being: The availability of many multilingual AI models and corpora usable as input for the task, generality of the task and ease of evaluation. These criteria are used in the next section to select concrete tasks for our implementation.

Model Availability in Many Languages

The goal of this work is to find approaches to find useful words for the purpose of language learning. Much research in Natural Language Processing is dedicated to improving NLP performance in English and other high-resource languages such as English, French or Mandarin Chinese [13]. This has the consequence that many AI models and other NLP methods achieve high levels of performance only in these languages, and many AI models are only available in English or only a small number of languages [13]. However, there are over 7,000 languages in the world [1], and for many of these there exist corpora, or online digital texts which can be used as inputs for our word utility evaluation approach, such as Wikipedia articles, *Open Parallel Corpora* (OPUS) [2], or *Wortschatz Leipzig* [3]. In order to make an implementation with the potential to exploit the linguistic diversity in these resources to the fullest extent, we only consider NLP tasks for which pre-trained models exist which can process a high number of languages.

Corpus Availability in Many Languages

The second point of consideration for task selection is **how much data is available for performing the task**. Ideally, we would like tasks for which suitable corpora are freely available or can be trivially generated from available corpora. This is because, with a larger amount of usable data, we not only improve the accuracy of our approach, but also increase the diversity of input data. Because context-specific language learning is a central motivation of this work, diverse training data is desirable, as it provides a broader range of linguistic contexts to model and from which to identify useful words.

[1] According to *Ethnologue* (`https://www.ethnologue.com/`, last accessed on April 22, 2025)
[2] `https://opus.nlpl.eu/`
[3] `https://wortschatz.uni-leipzig.de/`

Generality of Skill Required

Another important point to consider when selecting an NLP task is **how general the linguistic skills** are that the task requires: We use the performance in the NLP task as a proxy metric for the test subject's language ability. As such, we must ensure that task reflects a general level of semantic understanding, not only a narrow mechanistic skill that can be accomplished by using only a small part of the input. For this reason, we do not consider tasks such as part-of-speech tagging and text classification appropriate, as they do not reflect skills of language speakers used in everyday life.

Ease of Evaluation

To ensure we can measure task performance, we must also choose a task whose results can be easily compared with each other: Some tasks are difficult to evaluate objectively, or the evaluation takes an inordinate amount of computation (see Section 4.2.2 on text summarization). It follows that if we have the freedom to choose NLP tasks whose results can be automatically evaluated with good accuracy, we should choose them.

Summary of Desiderata

To summarize our desiderata for NLP tasks: Our implementation seeks to use NLP tasks for which AI models and Corpora exist in a large number of languages, to make our approach for finding useful vocabulary usable in the greatest number of languages. We prefer tasks that demonstrate general language understanding over tasks that only require a narrow skill set to perform or that are too technical, because general tasks are expected to align more with human linguistic skills. Finally, the task must be easily scorable, since the task score is the metric by which we gauge how useful words are. The next section introduces several candidates, primarily by surveying which tasks are commonly used as pre-training tasks for state-of-the-art language models, as these must fulfill similar requirements to those stated in this section.

4.2.2 Survey of Candidate NLP Tasks

This section introduces several NLP tasks we use in our implementation, as well as some tasks which were not selected. We first describe the process of how candidate tasks were found, and then go into detail for each candidate as to why it was or was not selected in the following subsections.

Candidates were first compiled by surveying common pre-training tasks which are used in state-of-the-art NLP models: This is because the purpose of a pre-training task is to endow the AI model with a general understanding of the language, before either using transfer learning to specialize it for a more specific downstream task, or using it before another downstream AI model to pre-process input [15]. Such tasks must necessarily be general and require general language understanding, since training the model with them is supposed to provide a solid basis for a wide variety of NLP tasks. Another benefit of using pre-training tasks is that their training is unsupervised, meaning there is no need to manually label data. Their

widespread use in Natural Language Processingalso means that pre-trained models are widely and freely available. From this consideration, we select language modeling and next sentence prediction as candidates, but reject language modeling for the reasons below. In addition to these common pre-training tasks also consider text summarization and sentence embedding, however, because text summarization is difficult to evaluate, reject this task as well.

Language Modeling

There generally exist two subtypes of language modeling, namely Causal Language Modeling and Masked Language Modeling [15]. Causal language modeling describes the task of predicting the text token in a text, given the sequence of previous tokens, and it constitutes the basis of many recent developments in Large Language Models [4] [27]. Masked language modeling involves filling in missing tokens in a sentence, and is one of the training tasks for the popular BERT models [17], along with next sentence prediction.

While its training data can be trivially generated from corpora and does not pose great challenges in evaluation, we did not consider either type of language modeling an appropriate NLP task for the purpose of finding useful vocabulary. This is because generating the task's training data already involves removing tokens from the input, resulting in training data with word distributions which do not correspond to language found in real texts. Another issue arises to the nature of model-agnostic, feature importance attribution XAI methods: These methods perturb the input to determine which tokens in the input are the most relevant. However, whether or not a token is appropriate as the next token in a document is sure to change with even slight perturbations: Consider the sentence:

The sky is blue.

A language model, given the input datum *The sky is*, may be able to predict that *blue* is a probable next word to this sentence.

However, if we mask the — relatively unimportant — word *is* in the sentence to find out if the model's can predict the last word of the sentence even when *is* is missing, we are left with the sentence:

The sky

For this variation, *blue* is no longer an appropriate next word to predict, as this would result in a grammatically incorrect sentence. This would lead a feature importance attribution method to evaluate *is* as a very important word in the sentence, because its absence leads to a loss of accuracy, despite it being unimportant to the overall meaning of the sentence. These complications make it difficult to use feature importance attribution methods on language modeling, which is why we decided against the use of this task for our implementation.

Next Sentence Prediction

In Next Sentence Prediction (NSP for short), the AI model takes as input two sentences and predicts a probability for the second sentence being the successor of the first sentence in their source text [17]. Input data for this task, including labels,

is trivial to generate from document-level corpora, as it merely requires a corpus of sentences that follow from each other, which is easily obtained from Wikipedia articles, film subtitles, or any other continuous text. As the output of next sentence prediction is one number expressing the predicted probability of the input sentences following each other, its performance is also easy to calculate, using cross-entropy. While predicting whether two sentences are consecutive may not seem like a transferable skill, it is used as one of two pre-training tasks for BERT models [17], which suggests that using this task in pre-training imbues the model with transferable language skills. For these reasons, we use the NSP task in our word evaluation approach.

Text Summarization

This task involves summarizing a given text, in other words, writing a shorter version of the input text while still conveying as much of the information from the original text as possible [29]. Summarizing texts accurately would seem to require a high level of understanding of the text, and thus be good choice for testing whether ablating certain words from the text has detrimental effect on the model performance. Unfortunately, this task is not appropriate for our purposes with respect to our other desiderata, because evaluating the quality of a summary is a very challenging task: It involves a ground truth summary which is usually manually created, against which another summary is compared. This already makes the use of this task problematic for low-resource languages (languages for which only few or only small NLP datasets exist), as text summarization datasets are scarce for these [5]. However, even assuming sufficient data, the comparison of two summaries is difficult: A standard way of evaluating summaries are ROUGE scores [2]:

These measure the overlap of n-grams (word sequences) between two texts. However, it is questionable how well they capture the similarity between texts, because they do not recognize the semantic similarity of synonyms, and a different sentence structure will result in a low ROUGE score even if the actual meaning of the sentences may be very close. For these reasons, we do not use text summarization in our implementation.

Sentence Embedding

Sentence embedding takes sentences as inputs and outputs vectors of numbers, which are typically used as inputs for a downstream task such gauging the similarity between two sentences, which can in turn be used to find similar sentences across languages for making training data for translations model [3] [30]. This approach can be performed on any corpus containing distinct sentences, meaning the corpus does not have to be document-level, and sentences need not be consecutive.

An important characteristic of this task is that it has no ground truth labels, and therefore, when using feature importance attribution XAI methods, we cannot measure differences in performance of the downstream task when the input is perturbed. Instead, we embed an input sentence in its original form (the baseline embedding), then embed variations of the sentence with some of its words removed, and measure the distance between the baseline embedding and the embedding of each variation. A large distance between the baseline and the variant embedding

where a word has been removed implies that the removed word is useful for understanding the sentence. We believe these distances are a valid metric for determining how much a word contributes to the meaning of the sentence, as the embedded vector is already a semantic representation of the input. In our estimation, the ease of evaluation and of finding input data, as well as its generality make sentence embedding an ideal task for word utility evaluation, which is why we use it not only for the generation of vocabulary lists, but also for the evaluation of their efficiency.

4.2.3 Choice of Model

In the previous sections, we have presented two tasks which seem appropriate choices for our vocabulary list generation and evaluation approach, namely, next sentence prediction and sentence embedding. This section discusses the choice of the AI model used for both of these tasks. In general, we use models from model families which support many languages, but the concrete models in our implementation are monolingual models, as these have shorter inference times, allowing us to process more data.

Next Sentence Prediction Model

As our next sentence prediction model, we use a BERT model [17], as these models have been trained in many languages (over 104 for `bert-base-multilingual-cased`, the model with the most languages [4]). As BERT models have the same structure to each other, this makes extending our approach to more languages trivial: We use the model-specific XAI method of attention (see Section 2.3.2) in our pipeline, the implementation of which is dependent on the transformer used, as the attention tensor's dimensions are dictates by variables such as the number of attention heads. However, for our implementation, we employ the monolingual English model `bert-base-uncased` [5] for its smaller size (110M parameters instead of 170M for `bert-base-multilingual-cased`) and shorter inference time.

Sentence Embedding Model

For the sentence embedding task, there are two model families that appear as suitable candidates for our implementation: *LASER* embedding models [3], developed by Meta, and *Sentence Transformers* a.k.a. *sBert* [30], developed chiefly by the University of Darmstadt. Both of these solutions have set as their aims the embedding of sentence of as many languages as possible, giving special considerations to low-resource languages.

While the latest *LASER* embedding model is more recent and claims to outperform previous models, its implementation requires a Unix environment to work, which presented compatibility issues with our Windows-based development setup [6]. Additionally, to the best of our knowledge, the models are not available on the *HuggingFace* [7] platform, meaning their use would have necessitated various changes

[4] `https://huggingface.co/google-bert/bert-base-multilingual-cased`
[5] `https://huggingface.co/google-bert/bert-base-uncased`
[6] `https://github.com/facebookresearch/LASER`
[7] `https://huggingface.co/`

in our implementation due the different interface. For this reason, we use the models from the `sentence-transformers` package. While the models are available in their dedicated Python package `https://www.sbert.net/`, these do not allow for the output of attention values, which is why we load the Sentence Transformer model from HuggingFace directly. This necessitates an additional post-processing step, but the code is available on the model website. The model we use is [8]. This model is a monolingual English model, however again, it is much more lightweight than the multilingual Sentence Transformer models.

4.2.4 Summary

We have put forth desiderata for the NLP tasks to make our approach to vocabulary list generation and evaluation accurate, as well as applicable to many languages and language contexts. As a result of these desiderata, we have chosen the two tasks of next sentence prediction and sentence embedding, for the general language skill they require, because they are easy to evaluate automatically, and because input data for them is trivial to acquire in many languages. The next section discusses the choice of corpora which are used as input data for these tasks.

4.3 Corpora

In the previous section, we put forth criteria for which NLP tasks should be used in our word utility evaluation approach, and decided on the two tasks of next sentence prediction and sentence embedding. Having decided on these, we now require data, i.e., corpora, as inputs for the tasks. Which corpora we use is another important decision, as these serve the purpose of modeling the language contexts in which the language learner is striving to achieve proficiency. This section therefore first states our general selection criteria for corpora. We then introduce publicly available corpora which are suitable for our evaluation approach, and conclude by choosing Wikipedia and OPUS subtitle data as inputs for the NLP tasks. From these data sources, we model linguistic contexts with two methods: A single article or subtitle, as well as a composite of various articles and subtitles For the background corpus used to calculate the inverse document frequency in TD-IDF, we use a corpus from the *Oscar* project (see Section 4.3.2).

4.3.1 Desiderata

This section puts forward our general criteria for corpus selection, following from our overarching goal of using the model to model linguistic contexts for the purpose of language learning. In short, we use corpora which are available in many languages, allow for the modeling of different linguistic contexts, and consist of continuous texts to provide data for the NLP tasks.

Document-Level Corpus

The first desideratum for our corpora stems from our choice to use next sentence prediction as one of our NLP tasks: As next sentence prediction predicts whether

[8] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

one sentence is likely the continuation of another, it requires contiguous sentences pairs to work. To generate these pairs, we must use (at least some) corpora featuring continuous tests, not only individual sentences. This presents a challenge, as many corpora which are compiled from Web Crawls, such as the corpora from *Wortschatz Leipzig* [9], contain only scrambled, single lines to avoid copyright infringements (single sentences cannot be put under copyright in certain states [9]). However, Wikipedia contains articles on many different topics which do not underlie strict copyright license (see Section 4.3.3), which makes it appropriate for our purposes.

Free Availability in Many Languages

In Section 4.2.1 we put forward reasons for why freely available AI models which can handle a diverse pool of languages are desirable for our undertaking. By the same token, we also prefer corpora which are publicly available in many languages over corpora which only include data in one language. While a monolingual corpus is not inferior to a multilingual one, its use necessitates that the user manually compile many corpora if a multilingual implementation is to be achieved.

Closeness to Linguistic Contexts Desired by Language Learners

As mentioned before, the corpus in our word utility evaluation approach serves the purpose of modeling a linguistic context, and this linguistic context should reflect some set of situations that a language learner is likely to find themselves in. Typical situations would include reading the news, reading literature or watching movies in their target language. As such, corpora which are close to the materials which language learners are likely to engage with are desirable, since their use makes the AI model's performance on the task more reflective of skills that a language learner would like to acquire.

Divisibility of Corpus into more Specific Corpora

Not only the relevance of the entire corpus's linguistic context is important: Some corpora enable us to further split them up into smaller corpora with more specific language contexts. For instance, while we can use a corpus such as *Wikipedia* as a whole, the structure of Wikipedia enables us to group articles by the category they belong to (politics/sports etc.), or even use single articles, to find more specific contexts. Such corpora are especially efficient for generating multiple contexts, hence we make use of such corpora.

Rejected Corpora

This section briefly describes two corpora which may appear to be useful for our word utility evaluation approach, but which we do not believe are appropriate.

Common Crawl: A common choice for training large language models is the *Common Crawl* dataset [10]. Its size and diversity could make it useful either as

[9] `https://wortschatz.uni-leipzig.de/`
[10] `https://commoncrawl.org/`

a corpus for modeling a "surfing the web" linguistic context, or as a background corpus to calculate TF-IDF on other corpora. However, we reject both uses of it: Firstly, in its raw form, it takes the form of HTML code and includes much boilerplate content such as cookie requests, necessitating much preprocessing. More importantly, speaking against its use as a context-modeling corpus, its data is not grouped by topic or language, which means we cannot trivially generate more topic-specific corpora from it, nor utilize it for different languages without a significant amount of pre-processing.

Wortschatz Leipzig: *Wortschatz Leipzig* [11] provides corpora in more than 100 languages, and groups these by their source such as "News", "Wikipedia", and "Web". We also rejected the Wortschatz Leipzig Corpora because, while these corpora are stripped of any HTML code, they are composed of single lines collected from websites, rather than whole documents. Although we considered using one of the "Web" corpora for calculating inverse document frequency for the TF-IDF metric, the original paper for the Wortschatz corpus also states that political content was utilized in bootstrapping the web searches for data gathering, including the *Universal Declaration of Human Rights* and the Jehova's Witnesses magazine *Watchtower* [12], which may lead to a respective bias in the final corpus [9]. Thus, we did not use *Wortschatz* corpora as a background corpus, either.

4.3.2 TF-IDF Background Corpus: Oscar

For the implementation of TF-IDF, we require a generic background corpus to normalize raw word frequencies found in context-specific corpora (that is, to calculate the IDF part of TF-IDF). For this purpose, corpora crawled from the internet offer themselves as an attractive option, as web content is diverse in that it encompassed both formal and informal content on many different topics.

We reject the Common Crawl and Wortschatz Leipzig datasets for the reasons already stated in Section 4.3.1. Instead, we use a corpus from the *Oscar* project [13] as a background corpus: Each *Oscar* corpus is a cleaned-up version of a *Common Crawl* dataset, on which preprocessing steps such as HTML stripping and removal of boilerplate content have already been performed. It consists of (whole) web pages, and most importantly, its content is language-tagged, covering more than 150 languages in total. There exist many versions of the Oscar dataset, but for this work, the *mOscar* [7] corpus from 2024 was chosen, both for its recency and coverage of many languages. Due to its vast size, we did not use the entire corpus, but a random sample of 53,178 documents from this dataset to calculate inverse document frequency.

4.3.3 Wikipedia

Our primary source for corpora are articles from Wikipedia [14]. Wikipedia is a valuable resource for several reasons: It contains a vast number of articles, and about

[11] `https://wortschatz.uni-leipzig.de/`
[12] Previously available at `watchtower.org`
[13] `https://oscar-project.github.io/`
[14] `https://www.wikipedia.org/`

a diverse set of topics. Wikipedia articles are assigned article categories describing their topic, such as "20th century American male actors". This means we do not have to group articles into topics ourselves, enabling us to trivially create corpora that model linguistic contexts more general than a single article, but more specific than Wikipedia as a whole. The characteristics of the article category system on Wikipedia are discussed below.

Wikipedia's articles are licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (*CC BY-SA*). Because of this, Wikipedia can provide dumps of its entire article database ready for download, with the articles in continuous form and with category metadata attached to them [15]. This enables us to use the articles as a source for the next sentence prediction NLP task.

While Wikipedia offers a large diversity of articles and article topics, the tone of language of the articles is generally academic. Therefore, Wikipedia is less likely to contain informal language, such as nonstandard language (English: *ain't*, *y'all*), slang, or language of a conversational form. This is especially relevant for languages such as Japanese, where grammatical forms in conversation vary significantly depending on the relationship between the speaker and listener: In Japanese, there exist various linguistic markers of politeness (*Teineigo* and *Keigo*) which play a crucial role in in-person interactions between Japanese speakers, as their presence or absence can signify respect, familiarity or humility with regard to one's interlocutor. However, such forms are rarely found on Wikipedia, as its articles do not take the form of a person addressing other people. Thus, we can see that texts on Wikipedia are of limited use to model linguistic contexts involving in-person conversion. To supplement for this lack of conversational language, we use another corpus composed of movie subtitles, which we present in Section 4.3.4. With these advantages and disadvantages in mind, the next section details our use of Wikipedia articles and their associated metadata to model linguistic contexts of various sizes.

Modeling Linguistic Contexts: One of the chief advantages of Wikipedia as a corpus is its diversity of covered topics. A language learner who is interested in a particular topic will likely find a matching article category on Wikipedia: For example, a person with an interest in American cinema who is learning English could likely take vocabulary from the articles of the category "20th-century American actresses" to boost their language learning in that linguistic context. For this reason, we use a number of manually selected categories as testing grounds for our list generation approach. For each category, we use a number of articles to make a list, then use different articles from the same category as testing data in the evaluation. We can thus check if learning vocabulary from a category genuinely prepares a learner in understanding texts from that context.

At the same time, a learner could also wish to make a vocabulary list from concrete Wikipedia articles they would like to read. For that reason, also use single articles as corpora to be processed by our list generation approach. In that case, no split of training and test data is necessary, as the aim is not to model general knowledge about a domain, but equip the learner with vocabulary about the **closed** context of a Wikipedia article. Therefore, in the single-article approach, both the

[15] Available for download at `https://dumps.wikimedia.org/`

list generation and evaluation use all lines of the article.

Pre-Processing: In order to use Wikipedia articles as inputs to our word utility evaluation approach, we perform pre-processing on the raw data to filter out source code, as well as chapters that we do not consider to be part of the article text. This section shortly describes this process.

In their raw form, Wikipedia dumps are written in a markup language called *WikiCode*. To filter this code and obtain only natural language, we employ the *mwparserfromhell* package [16] to first strip the text of hyperlinks and markup, and also manually remove metadata such as article categories from the article text. Next, we filter out the text those sections under headings from a hard-coded list. This list was compiled by the author and can be seen in Table 4.1. The reason for their exclusion is that these chapters are pure lists which do not contain full sentences, and thus unlikely to contribute to modeling a linguistic context.

Section Titles
References
See also
External links
Further reading
Notes
Bibliography
Filmography
Discography
Published works
Sources
Citations

Table 4.1: Common non-article sections in Wikipedia articles

Finally, we split the article into individual sentences with the *NLTK* python library [17]. Which these pre-processing steps, we attain a corpus which mostly contains full sentences which describe the article's topic.

4.3.4 OPUS OpenSubtitles Parallel Corpus

The previous section has presented Wikipedia as a source of input data which can be used to model many different linguistic contexts, especially on many topics, but is not very diverse with respect to the tone of writing. This section will introduce a type of corpus which includes more informal language, as it is composed of movie subtitles. The OPUS collection [18] is a collection of so-called parallel corpora: Corpora which have text segments in one language aligned with the presumed translation of the segment in a second language. OPUS corpora have been used to train machine translation models such as OPUS-MT [36], a freely available set of transformer models for translation, including between low-resource languages. In this work, we make use of only the English sentences in one such parallel corpus, as we do not

[16] `https://github.com/earwig/mwparserfromhell`
[17] `https://www.nltk.org/`
[18] `https://opus.nlpl.eu/`

require their translations. Rather, our reason for using corpora the OPUS collection is that one of its corpora collections uses movie subtitles as its source and can thus model the linguistic context of watching movies, a popular pastime:

One of the largest collections of corpora inside OPUS is the OpenSubtitles dataset [19]: Its sentences are generated from subtitles from the popular subtitle sharing platform *OpenSubtitles* [20], but are pre-processed in numerous ways to ensure their integrity and usability as parallel sentences [22]:

- In their original form, movie subtitles are stored in SRT format, which does not save sentences, but so-called subtitles blocks. These are the segments which appear on-screen when watching the movie with its subtitle, including both text and start and end time of the block appearing on-screen. One such block may contain multiple sentences, or only a partial one, meaning there is an n-to-m-relationship between subtitle blocks and sentences. However, the lines which make up the OPUS OpenSubtitles dataset are split and joined such that each line (ideally) contains one sentence. We found that this is not always the case, but exceptions are sufficiently rare to not necessitate further pre-processing.
- The SRT files may contain spelling errors. These are due in part to human errors, but may also occur when a subtitle file is generated from scanning a paper document and performing Optical character recognition. Spelling errors are checked and corrected to some extent in the OPUS OpenSubtitles dataset.
- A movie may have different subtitles in one language associated with them on OpenSubtitles. In the OPUS OpenSubs dataset, available subtitles are compared in order to identify the subtitle pair which is most likely to be accurate in its alignments and free from errors such spelling, taking into account metadata such as user ratings of subtitles. Then, the subtitle pair which aligns the best is chosen to appear in the OPUS dataset. This corrects for issues which would otherwise be found in the dataset, such as poor translations.

The full process, as illustrated by the authors, can be seen in Figure 4.1 As of 2025, the latest version of the corpus (v2018) contains aligned subtitles of 62 languages between each other.

The OPUS Subtitles dataset contains not only the parallel sentences, but also movie identifiers from the Internet Movie Database (*IMDB*) [21], which enables us to reconstruct the movie from whose script each sentence originates. While the dataset contains contiguous subtitles, these are not appropriate input data for the NSP task, as film subtitles consist of dialogue instead of continuous text. Our preliminary tests showed this, as our NSP model showed very low performance when given contiguous subtitle sentences pairs as inputs.

[19] `https://opus.nlpl.eu/OpenSubtitles/corpus/version/OpenSubtitles`
[20] `https://www.opensubtitles.org/`
[21] `https://www.imdb.com/`

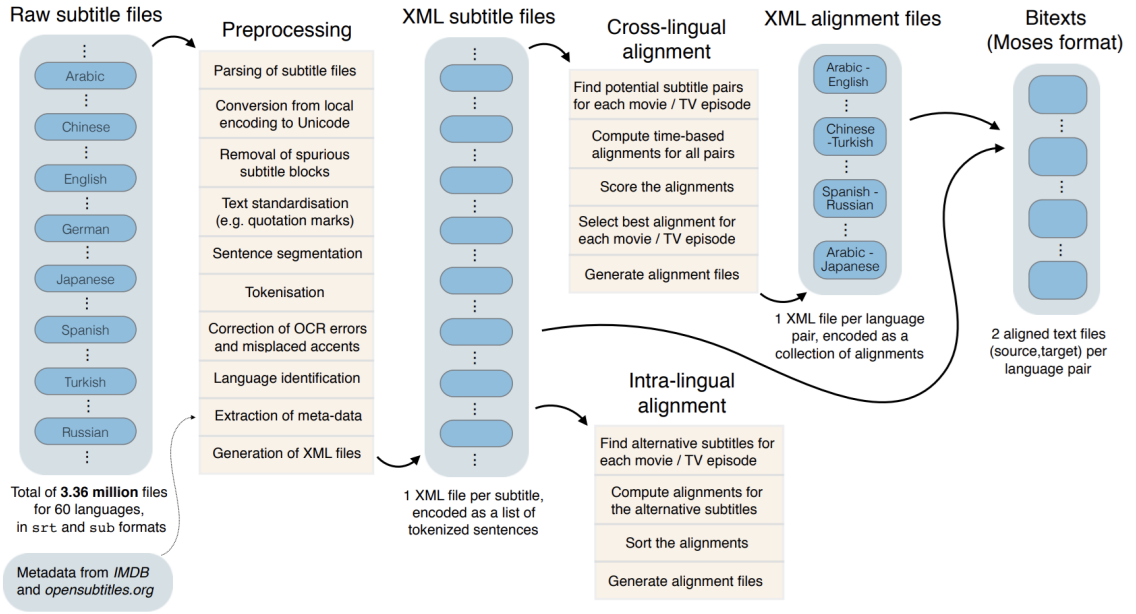

Figure 4.1: Pre-processing of the OPUS OpenSubtitles parallel corpus.

4.3.5 Construction of NSP Corpora

As mentioned in Section 4.3.1, the next sentence prediction task requires consecutive sentences in order to be performed. Out of the two context-specific corpus types which we employ, only Wikipedia articles consist of such consecutive sentences. This is confirmed by the fact that preliminary investigations showed that the NSP model's performance on OpenSubtitles corpora was essentially equal to that of random guessing. Thus, the NSP task is only performed on Wikipedia articles. The NSP corpus of an article is constructed by simply taking each consecutive sentence pair from the article, constituting an input datum for the NSP model with an `is_next` label of `True`. We do not construct negative examples, for the reason that the most words with the most impact on the performance of the model would be words that tell the model that two sentences are *not* connected, which does not appear to us to be useful information.

4.3.6 Final Data Set

We have described the sources of the corpora used in the implementation of our approach, but from these sources, we test our approach on two scales: We call the first test scenario a **small context** approach, where we generate a vocabulary list from a single Wikipedia article or OPUS subtitle. We then evaluate the generated vocabulary list on the same corpus. This covers the intended use case where a learner wishes to read an article or watch a movie, and study the most important words beforehand. We call a single article or single subtitle a **single corpus** from here on, to avoid repetition.

In the second scenario, which we call **large context**, we generate the vocabulary list from a large number of single corpora, grouped by context, and the set of single corpora used for generation evaluation is disjoint: For Wikipedia articles, we use 80% or the articles in each Wikipedia category to create a vocabulary list specific

to that category, whereas for the OpenSubtitles dataset, a sample of 632 subtitles is used. Evaluation, meanwhile, is performed on the remaining 20% or articles for each Wikipedia category , and a set of 30 randomly chosen subtitles (not used for list generation) for the OpenSubtitles dataset. The large context scenario was created to test how useful the words given by each approach are to understanding a large context, where a learner is trying to increase their general understanding of texts in a certain context, without knowing the exact texts before engaging with them. We can thus categorize our evaluation by two both the corpus source and the scale of the evaluation, giving us four categories:

1. A single Wikipedia article, where the lines used for list generation and evaluation are the same.
2. A single subtitle, where the lines used for list generation and evaluation are the same.
3. A Wikipedia category, where vocabulary lists are generated on the "train" set of articles in that category, and evaluated on the "test" article set. The proportion of "test" articles s 20% .
4. A large sample of subtitles, with designated "train" and "test" subtitles as well. Due to the large number of subtitles, we were not able to include the entire dataset in the list generation.

Table 4.2 shows the number of single corpora (articles or subtitles) in each composite corpus.

Composite Corpus	Train (List Generation)	Test (List Evaluation)
1980s English-language films	20	6
20th-century American actresses	44	11
20th-century American male actors	86	22
World War II political leaders	16	5
OpenSubtitles	632	30

Table 4.2: Number of "train" and "test" corpora in each composite corpus.

To gain a better understanding of the data, Table 4.3 shows a few key metrics of each corpus type. It can readily be seen that, while the corpora taken from the OpenSubtitles subtitles are composed of more lines on average, the Wikipedia articles feature much longer sentences (over three times longer on average than the sentences in subtitles). This makes sense, as Wikipedia articles are written in an academic style, while movie subtitles tend to be more informal. Wikipedia articles also contain a higher number of unique words on average, despite containing a lower number of words in total. This may hint at a higher diversity of vocabulary employed in articles, although a higher number of named entities may also contribute to the higher count of unique words.

For a more granular overview, we also show these metrics for individual categories in Table 4.4. Most obvious is the fact that the articles of World War II political leaders are, on average, much longer than those films or actors of either

Corpus Type	# Corpora	Lines	Words	Unique Words	Words per Line
opensubs-subtitle	632	901	5436	1050	6.0
wikipedia-article	210	253	4382	1211	17.3

Table 4.3: General statistics on corpora, per corpus type.

gender. Another notable pattern is that the articles of male actors tend to be longer than those of their female counterparts.

Wikipedia Category	# Corpora	Lines	Words	Unique Words	Words per Line
1980s English-language films	26	160	2950	1014	18.4
20th-century American actresses	55	188	3296	934	17.5
20th-century American male actors	108	245	4157	1180	17.0
World War II political leaders	21	575	10159	2340	17.7

Table 4.4: General statistics on corpora, per Wikipedia article category.

4.4 XAI Methods

In order to appraise the influence of words in a corpus on the performance of an AI model performing NLP tasks on it, this work employs Explainable AI methods, as explained in Section 3.5. This section first puts forth the desiderata for the XAI methods, and then presents our choices of methods. Due to time constraints, we limited our research to relatively simple methods. While we gained valuable insights with the methods presented in this chapter, we acknowledge that better results may be achievable by using XAI approaches with more robust research.

4.4.1 Desiderata

Section 2.4 has introduced essential distinctions between XAI methods which help us determine which methods are appropriate for our purpose: We employ local, feature attribution methods of both model-specific and model-agnostic type, as these allow use to gauge the importance of words in the input from multiple points of view. Model-specific methods have an advantage in terms of computational efficiency, as they generally require fewer model calls to attribute importances to features, instead leveraging the internal structure of a model to arrive at explanations. On the other hand, model-agnostic methods are useful in that they can be used on any model and gauge the input-output relationship by performing actual experiments on the model, by observing the way changes to the model input change its output.

There is another desideratum which we impose on our XAI methods: The XAI method should allow for processing the input sentences in its original form and not only as a *bag of words*: Some methods, such as SHAP [23] and LIME [31], were originally developed for structured data with a fixed number of input features. When they are used on NLP tasks, they typically use as inputs not the textual data in its raw form, but rather vectors indicating whether a particular word is present in the sentence, with no regard to the word order and thus of syntactic relations between

words. In our view, this is problematic, as simplification of the input could lead the methods to attribute low importance to words such as "not", which only affect the meaning of a sentence in combination with its other words. For this reason, we did not employ the popular XAI methods SHAP or LIME in this work.

In the following sections, we present three model-agnostic XAI methods which we use in the implementation of our vocabulary list generation approach described in Section 3.5. These work by perturbing the inputs to the model and analyzing the changes in output. The difference between them is how the perturbation is done: Single Token Ablation removes single tokens from the inputs, whereas Single Token Summary works by setting the input to a single token. The third method, Progressive Summary, attempts to find the most important words in sentences by iteratively building up the input sentence, one token at a time. While the sections make reference to changes in "performance" to calculate word utility, this is only strictly accurate for the next sentence prediction task. In the case of the sentence embedding task, we measure performance of the model on a permuted sentence by how close the output embedding vector is to the embedding of the unpermuted sentence, as explained in Section 4.2.2. After the model-agnostic methods, we detail our use of the model-specific transformer attention approach mentioned in Section 2.3.2, as it requires more post-processing than the model-agnostic methods.

The relationship of the XAI methods to the NLP tasks is also discussed, as not every combination of XAI method and NLP task is feasible, either because it requires an unrealistic amount of computation or because the results of the combination are difficult to evaluate. The results of the experiments with these methods is presented in Chapter 5.

4.4.2 Single Token Ablation

Our aim in using XAI methods is to find out which words, if known, improve the model's performance the most, when compared to not knowing the word. Perhaps the simplest way of testing this is by going through every word in the AI model's input, and examining the impact on model performance when the word is removed from the input. Within this work, this approach is called **Single Token Ablation**. The processing steps performed in Single Token Ablation can be seen in Figure 4.2.

Approach: For an input sentence, we run the model on the unmodified sentence to get a baseline performance. We then get the unique words in the input by using the tokenizer and merging sub-word tokens back into words. For each word in the input, we create a variation of the input where the word is removed. We then run the model on all variations and measure the performance. The estimated utility in this approach is then calculated by taking the difference in performance between the variation's performance and that of the baseline output.

Critical Evaluation: One advantage of Single Token Ablation is its conceptual simplicity, as it is easy to implement. Furthermore, its time complexity is fairly low for a feature method, as it requires only one model call for each unique token in the sentence ($\mathcal{O}(n)$). Before running experiments, we anticipated that due to this simplicity, Single Token Ablation would not work well on long sentences, where

leaving out a single token would not seem to have a great impact on the meaning of the sentence. Furthermore, this method does not take into account the utility of a word in the context of word combinations. However, among the methods used in this paper, this method achieves the best results in the final evaluation.

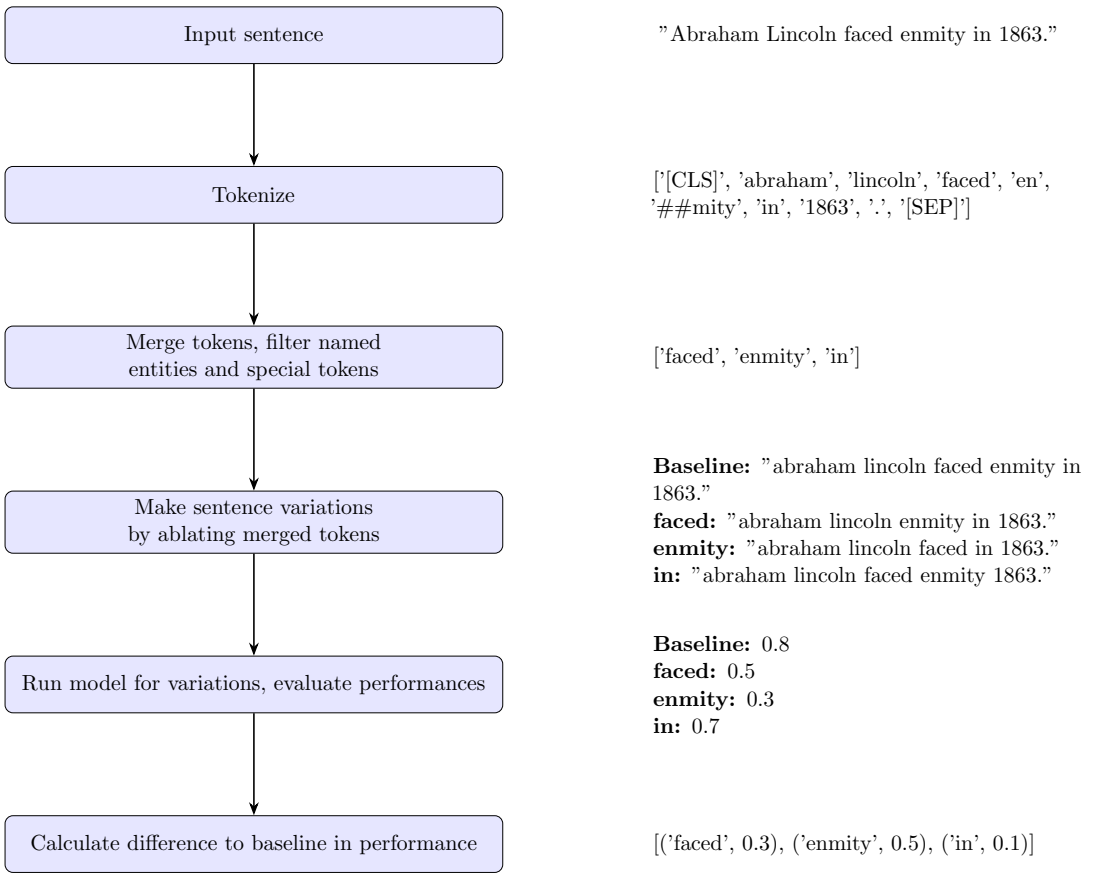

Figure 4.2: Example of Single Token Ablation, on one sentence.

4.4.3 Single Token Summary

The second feature model-agnostic method can be seen as the opposite of Single Token Ablation: Instead of taking out one word at a time, we use as input only a single word at a time. Because the intention behind this method is finding words which, on their own, carry much of the meaning of a sentence, we call it **Single Token Summary**.

Approach: We first run the model on a variation of the input sentence where only punctuation and named entities are included, using the output as the starting performance. We then decompose the input sentence into its word tokens. For each word, a variation of the sentence is put through the model where, in addition with punctuation and named entities, only that word appears. The utility in this approach is calculated by taking the performance improvement of a word's sentence variation compared to the starting performance.

Critical Evaluation: Like Single Token Ablation, this method requires only one model call for each unique token in the sentence ($\mathcal{O}(n)$). This method may find

tokens helpful to beginners more efficiently, however, as it simulates knowing only one word, whereas Single Token Ablation tests performance while taking into account the meaning of the other words in each sentence, as well. However, this could the method to assign low utility to words which mostly convey meaning when in combination with other words, such as "not".

4.4.4 Progressive Summary

While Single Token Summary is easy to implement and computationally inexpensive, it has the conceptual disadvantage of not taking into account the utility of word combinations. To achieve a more accurate evaluation, we present another "summary" approach, where the sentence is built up one word at a time. We call this approach "Progressive Summary".

Approach: For an input sentence, this approach starts the same as Single Token Summary, by checking the performance of the one-word sentence variations. We then rank the words by their performance improvements, and use the word with the highest improvement in the next iteration. From the remaining words in the sentence, we check which one improves the performance the most when used in addition to our already found word. By iterating this method, we progressively reconstruct the original sentence, which each step adding the word which improves the performance the most. Here, the estimated utility of a word is the boost in performance which is brought to the sentence when the word is added.

Critical Evaluation: For a single sentence, Progressive Summary would appear to be the most accurate in ordering its words by utility, as it considers the utility of each word in combination with the words which have already been found to be useful. However, Progressive Summary is much more computationally intensive than Single Token Summary, which requires only one model call per word in the sentence. Because of its iterative approach, for a sentence with 10 words, Progressive Summary calls the model $10 + 9 + 8 + ... = \sum_{i=1}^{n} i = \frac{n^2+n}{2}$ times, which is a time complexity of $\mathcal{O}(n^2)$.

Compatibility with NLP tasks: As Progressive Summary starts from a set of one words and then progresses to larger sets, we do not consider it appropriate to employ in combination with next sentence prediction. One could circumvent the issue by leaving one of the sentences in its original form and only perturbing the other, but this can hardly be seen as a modeling of natural language ability. As such, our implementation does not use Progressive Summary with next sentence prediction.

Summary: Progressive Summary attempts to find the most important words in a sentence one by one, by first reducing the sentence to one word, then two etc., and using a distance metric to compare the "summaries" to the original sentence. The words which bring the model output for the summary closest to that of the original sentence are regarded as the most useful.

4.4.5 Attention as Explanation

The model-agnostic methods so far could be employed on any AI model. However, as many of the latest state-of-the-art AI models follow the Neural Network architecture, and more specifically the transformer architecture (see Section 2.3.2), we also attempt word utility estimation with method specific to these models, namely attention. As stated before, transformer models use self-attention to assign importance to the connections between tokens in the input (if their input is in text format). This may be used as a kind of explanation, as words which have strong connections to many other words in the sentence presumably are more important to the AI model's understanding of it than those receiving less self-attention.

An important difference to the model-agnostic methods explained before is that we do not control word masking with a tokenizer *independent of the model*, thus the splitting of words is not under our direct control. The tokenizers used are `bert-base-uncased` and `sentence-transformers/paraphrase-MiniLM-L6-v2`. These produce tokens on a sub-word level, which means that we must perform post-processing on the attentions to acquire values on the word level and make vocabulary lists than contain full words. For BERT tokens, this is a simple process as sub-tokens which are not at the beginning of a word start with "##" (See Section 2.2.3). The estimated utility for the reconstructed token is calculated by taking the sum of the attention values of each sub-token.

Approach: A transformer model is run with one line as input. The transformer outputs an attention tensor, which attributes a value to each pair of tokens in the input (this includes special tokens from the tokenizer, such as `[SEP]`, and punctuation). We then aggregate the attention of each token by summing over one axis of the attention tensor of all connections each token has, and over all the attention heads, resulting in one scalar value per token. The original tokens of the tokenizer are merged Next, special tokens and punctuation are filtered Finally, we normalize the attentions such that the attention for all words in the line sums up to one. The attributed attention to a word is its estimated utility.

Critical Evaluation: As hinted at before, this method uses one model call for one input line, and thus has a time complexity of $\mathcal{O}(1)$. One disadvantage of attention as our explanation mechanism is that attentions are attributed to input tokens of the model. Unlike the model-agnostic methods where word masking is controlled by a tokenizer *independent of the model*, the tokenizer cannot be changed. AI models are trained with some tokenizer as preprocessing the input, and this tokenizer cannot be changed, lest we change the input format to the AI model itself. This is an issue because it means that lists generated with the model-specific tokenizer are not necessarily comparable to lists from model-agnostic methods, as the tokenization method may differ. In addition to this technical downside, attention is controversial as a mechanism to explain model decisions: One point of critique is that, while attention influences the model's behavior, it is not a faithful explanation because alternative attention distributions can yield similar model outputs [12]. On the other hand, it has been argued that the mere fact that a model chooses a particular attention distribution, rather than another, means that the actually generated distribution is a more plausible explanation that artificial adversarial distributions [40]. As we

will show in our list evaluation, when used with our list generation approach, attention produces vocabulary lists that are close in efficiency to those produced by the model-agnostic methods.

Summary: Among the XAI methods used in this work, Attention as Explanation requires the least resources, and it is not restricted to any one NLP task (although the model must be of transformer type). However, we do not control the tokenizer used directly. While the NSP model uses the same tokenizer that is also used for token masking in the model-agnostic XAI methods, the sentences embedding model uses a different tokenizer.

4.5 Implementation with Chosen NLP Tasks, Corpora, and XAI Methods

The previous chapters have argued for the choice of components in our theoretical approach to vocabulary list generation and evaluation. This final section about our implementation describes how the chosen components interact in our implementation.

ToDo: Describe which comps are compatible, and where it is not obvious, how exactly they are made to work with each other.

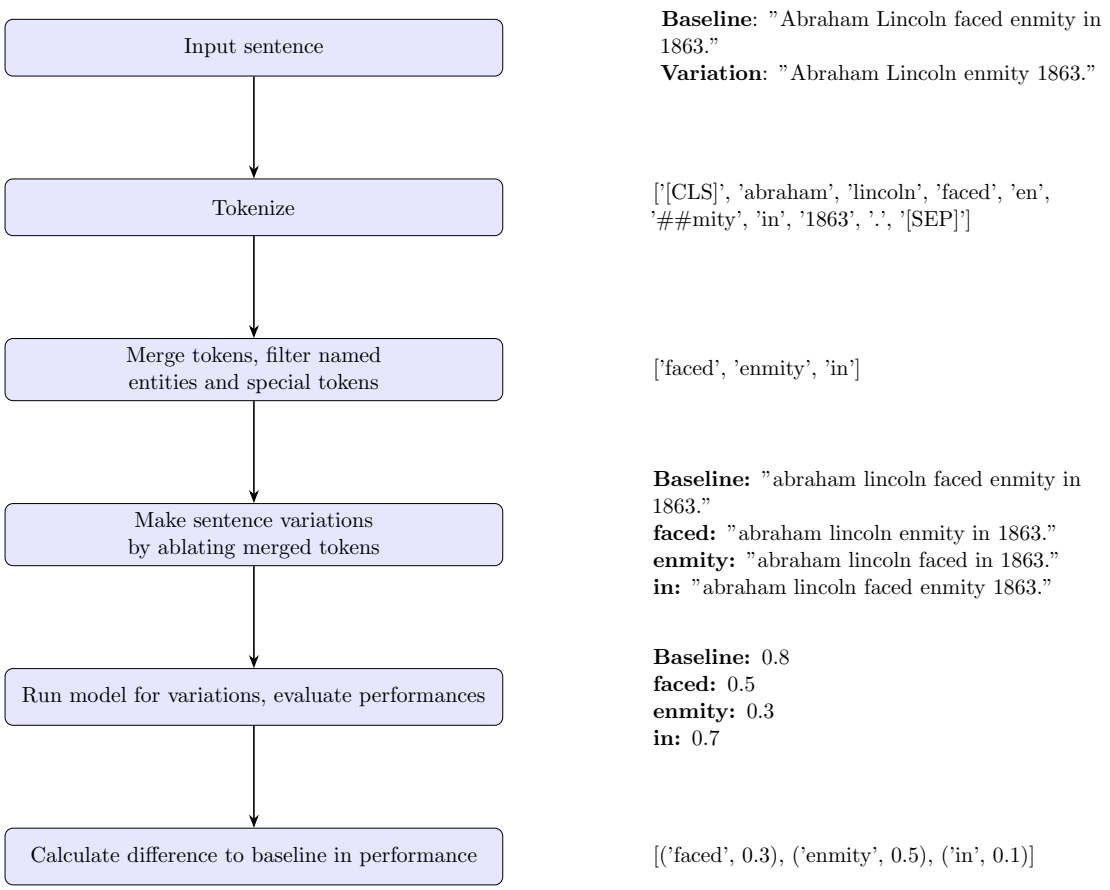

Figure 4.3: Example of sentence embedding scoring

4.5.1 Supported Languages

Our data pipeline contains various language-specific components, all of which must support a language in order for our approach to support vocabulary list generation in the language. Table 4.5 summarizes the language support of each component, including pre-processing.

Component	**Supported Languages**	**Notes**
Tokenization	104	Using the `bert-base-uncased` model[1]
Named Entity Recognition	9	Using the *spacy* model `xx_ent_wiki_sm`[2]. The number of supported languages is not stated for this model, but it was trained on the `WikiNER` dataset[3], which contains 9 languages.
Next sentence prediction	104	Using the `bert-base-multilingual-cased` model[4]
Sentence Embedding	> 50	Using the `paraphrase-multilingual-MiniLM-L12-v2` model[5]
Sentence splitting (for Wikipedia articles)	> 150	Using the `xx_sent_ud_sm` model[6], based on the *Universal Dependencies* framework[7], which supports > 150 languages.
Filtering of Wikipedia sections	(manual)	See Section 4.3.3
Wikipedia Corpus	> 300	As of 2025, there exist Wikipedias in over 300 languages[8]
OPUS OpenSubtitles Corpus	94	See [9]

Table 4.5: Languages supported by each component in the pipeline.

It can be seen that, apart from the manual selection of certain Wikipedia headings, named entity recognition is the bottleneck in our pipeline with the currently used packages. While we did not individually cross-correlate the supported languages of each component, we estimate that the pipeline supports 9 languages when taking into consideration the *spacy* NER capabilities. With a NER model supporting more languages, the implementation is estimated to support at least 40 languages, as the next bottleneck would be the sentence embedding model with over 50 languages, some of which may, however, not be supported by all other components.

[1] https://huggingface.co/google-bert/bert-base-uncased
[2] https://spacy.io/models/xx#xx_ent_wiki_sm
[3] https://huggingface.co/datasets/mnaguib/WikiNER
[4] https://huggingface.co/google-bert/bert-base-multilingual-cased
[5] https://www.sbert.net/docs/sentence_transformer/pretrained_models.html#original-models
[6] https://spacy.io/models/xx#xx_sent_ud_sm
[7] https://universaldependencies.org/
[8] https://en.wikipedia.org/wiki/List_of_Wikipedias
[9] https://opus.nlpl.eu/OpenSubtitles/corpus/version/OpenSubtitles

Chapter 5

Evaluation

Evaluation of the vocabulary list generation approaches is challenging: The accuracy of the approaches is evaluated by evaluating the efficiency of the vocabulary lists. We have described in detail our approach to vocabulary efficiency evaluation in Section 3.3, which involves running an AI model on a context-specific corpus with a given vocabulary, where the vocabulary is given by taking the first n k words from the vocabulary list in question. However, this approach is very similar to our approach for list generation, which involves analyzing AI interaction of AI models with corpora through the use of XAI methods. Therefore, it is natural to assume that our evaluation approach is biased towards the lists stemming from our list generation approach. This concern is especially pertinent as it has been shown that AI models do not reliably respond comparably to humans when the inputs to NLP tasks are perturbed [37].

To the best of our knowledge, there has not yet been a proposed evaluation metric for vocabulary lists that can be done automatically and which outputs a simple numeric result. For this reason, in addition to our own evaluation approach utilizing AI models, we take some sample vocabulary lists, use our evaluation approach, and check whether the results align with human intuition.

For this reason, our evaluation is not only concerned with analyzing statistics of measured efficiency, but also with whether the calculated efficiency values are reliable. The questions examined in this chapter can be summed up as:

1. Does the efficiency evaluation of our list correspond with human intuition?
2. Does our generation approach outperform frequency-based approaches?
3. What are the upsides/downsides of each list generation approach (accuracy, speed, free choice of corpora)?
4. How can the various approaches be combined to supplement each other's weaknesses?

ToDo: Add section references when chapter structure is fixed The following sections first describe the (numeric) metrics used in the evaluation. We then introduce several baselines against which we compare our list generation approach. The baselines include both publicly available vocabulary lists from online language

learning tools, and existing list compilation methods introduced in Section 2.5.2 in connection with the corpora used for our own lists for a more direct comparison.

We then present the results of the evaluation, as well as discuss their implications.

5.1 Evaluation Measures

The various utility extraction approaches produce ranked lists as outputs. To compare these, we employ both quantitative and qualitative comparison approaches, described in the following sections.

5.1.1 NLP Task Performance

Our primary metric for evaluating a vocabulary list's efficiency is the approach explained in Section 3.3: We let an AI model perform an NLP task with a vocabulary $V_{l,k}$, consisting of the first k elements of the list l. The limited vocabulary is simulated by removing words outside the vocabulary from the inputs to the model. For illustration, consider the example sentence:

Abraham Lincoln faced enmity in 1863.

If our vocabulary consists only of the word *enmity*, this sentence becomes:

Abraham Lincoln enmity 1863.

Since *Abraham Lincoln* is recognized as a named entity, we leave it in the input, even though it is not part of the vocabulary. Non-alphabetic tokens, such as numbers and punctuation, are also left in the input.

We start the evaluation of every vocabulary list with a vocabulary size of 0, letting the model run on inputs where only named entities and non-alphabetic are left. The score for the entire corpus is calculated by taking the arithmetic mean of the scores of individual lines. This allows us to see the "utility" of the named entities on their own, putting into comparison the following performance scores in our evaluation. After this, we set the vocabulary size to 10 words, and run the AI models with the resulting vocabulary $V_{l,10}$. We then progressively increase the vocabulary by increasing index k until we reach the end of the list. The increase of vocabulary size is exponential, as this ensures a quick evaluation and performance is expected to not grow very much towards the end of the list, where the less useful words should be (this assumption can be confirmed in the evaluation results in Section 5.3).

Listing 2 shows pseudocode for the described setup:

ToDo: Edit surrounding descriptions of efficiency to be in line with pseudocode

ToDo: Explain that a performance score between 0 (worst input) and 1 (unaltered input) is used for normalization.

This approach gives us a direct, quantitative metric for the efficiency of a vocabulary list.

Algorithm 2 List Efficiency Evaluation.

Require: voc_list, corpus, model
1: $INIT\_VOCABULARY\_SIZE \leftarrow 10$
2: $VOCAB\_EXPANSION\_FACTOR \leftarrow 4$
3: $vocabulary\_size \leftarrow VOCABULARY\_SIZE\_INIT$
4: Initialize $vocabulary\_scores \leftarrow$ empty list
5: **while** $vocabulary\_size < length(voc\_list)$ **do**
6: $vocabulary\_size \leftarrow vocabulary\_size * VOCAB\_EXPANSION\_FACTOR$
7: $vocabulary \leftarrow voc\_list$ until index $vocabulary\_size$
8: Initialize $line\_scores \leftarrow$ empty list
9: **for** each $line$ in $corpus$ **do**
10: $line\_with\_only\_known\_words \leftarrow line$ without words not in $vocabulary$
11: $baseline\_output \leftarrow$ model($line$)
12: $new\_output \leftarrow$ model($line\_with\_only\_known\_words$)
13: $line\_score \leftarrow$ similarity($baseline\_output$, new_output)
14: Append $line\_score$ to $line\_scores$
15: **end for**
16: $vocabulary\_score =$ sum ($line\_scores$)
17: Append ($vocabulary\_size$, avg_score) to $scores$
18: **end while**
19: **return** $scores$

As an illustration, we present the results of one such efficiency evaluation run on a corpus can be seen in Figure 5.1.

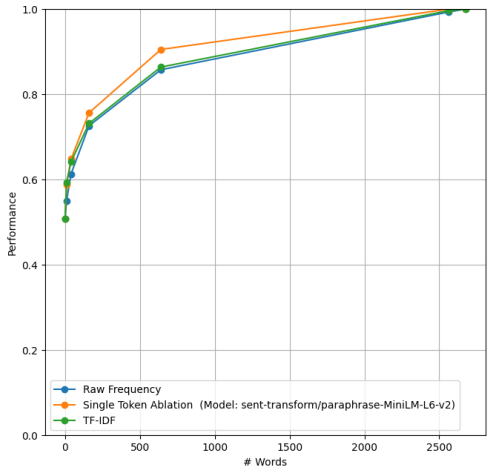

Figure 5.1: Efficiency evaluation results on vocabulary lists for the Wikipedia Article of American actor Marlon Brando [1].

To make the figure readable, only three list generation approaches are compared, namely Raw Frequency, TF-IDF, and Single Token Ablation with the sentence embedding model. We first evaluate the AI model performing the task with

[1] `https://en.wikipedia.org/wiki/Marlon_Brando`

a vocabulary size of zero, simulating the state of not knowing any words. The only tokens in this run are named entities and punctuation, as these are not considered to be potential vocabulary terms. The vocabularies are then evaluated with an initial size of 10, and increase by a factor of 4 in each step, until a final evaluation is done with the full vocabulary list. The full list typically contains all words from the corpus, which is why the performance reaches 1 with all lists. To compare the results on a broader scale, we aggregate the results of the evaluation runs for each corpus type, word utility evaluation method and AI model (where applicable) by taking the average performance of the generated vocabulary list, with vocabulary sizes 0, 10, 40, 160, and 640.

Utility Aggregation for Large Context Scenario

ToDo: the following is supposed to describe aggregation both for gen and eval. Make it understandable. As described above, the utility of a word for a given single corpus is calculated by taking the sum of the word's utility across the corpus' lines For the large context evaluation scenario, we must also aggregate the utilities across corpora. The simplest method would be to simply take the sum of each word's utility across all lines in the composite corpus. However, this approach is biased against small single corpora, which contain fewer lines and are thus assigned less total word utility. For this reason, we normalize the word utilities such that each corpus contributes the same total word utility. For the evaluation too, we average over the performance scores of a list across corpora instead of across lines.

When we find two approaches for list generation produces vocabulary lists of similar efficiencies, we might ask ourselves if these lists are similar: Can two lists which differ substantially in their ordering produce similar performance? To answer this question, the next section introduces additional evaluation metrics which measure the similarity of two lists.

5.1.2 List Similarity

Apart from our own vocabulary list evaluation approach that measures performance of an AI model using the vocabulary list, we also compare how similar the lists are in general. In order to compare the similarity of the resulting vocabulary lists provided by our approaches, metrics are needed that can be consistently calculated across the different approaches. While a human may be able to qualitatively analyze lists and gain a rough idea of their similarity, computed metrics provide an instantaneous (if simplified) outlook on similarities. A metric shall be defined as a function which takes as parameters two word lists of equal length which are word lists ordered in descending order by supposed utility, and outputs a real number giving either a distance or similarity between the lists.

Considerations of the choice of metric are:

Handling of lists with partial overlap. Metrics must be able to handle elements which occur in only one of the two lists. Thus, a metric which solely compares ranks of elements is not viable.

Start of lists is more impactful than end. Since the beginning of lists contains the words which are ranked as most important, changes at the top should

impact the metric more than changes at the bottom. This includes (1) Equal differences in rank should be counted as more important if they occur further up the list. A word that is rank 1 in list A, but rank 101 in list B says more about the similarity than if a word is rank 2000 in list A, but rank 2100 in list B. Likewise, if a word is absent from list B, it implies a greater difference if that word is at rank 1 in list A than if it were at rank 1000.

Metrics Used

Sequential rank agreement (modified) [6]: This metric is based on the deviations of some subset of the lists in the upper ranks. It is important to note that this metric has an additional parameter "depth" which determines how many elements (from the top of the list) are considered. It is therefore more helpful to view its results at various depths. The original formula for this metric in the case of two lists is:

$$a_d := a \text{ from start to rank } d$$

$$S_d := a_d \cup b_d$$

$$SRA_d(a,b) := \lambda \cdot \frac{\sum_{x \in S_d} \sigma^2 \left((r_b(x)) - (r_a(x)) \right)}{|S_d|}$$

where λ is a normalization factor ensuring that $\max(SRA) = 1$. In its proposed form, this metric can only compare lists which contain the same set of unique elements, just in different orders. In order to make it work on lists where this is not the case, one can set the "rank" of nonexisting elements to a value greater than the length of the lists, such as $2|a|$. Another drawback of the metric is that the standard deviation of two numbers does not depend on their absolute value, only their difference. However, to satisfy number 3 of the stated requirements, we can take the deviation of the logarithm of the ranks instead of the deviation of the ranks themselves, resulting in the formula

$$r'(x) := \begin{cases} \text{rank}_b(x) & \text{if } x \in b, \\ 2 \cdot |a| & \text{otherwise.} \end{cases}$$

$$SRA_d^{mod}(a,b) := \lambda \cdot \frac{\sum_{x \in S_d} \sigma^2 \left(\log(r'_b(x)) - \log(r'_a(x)) \right)}{|S_d|}$$

For this modified version, λ can be calculated with:

$$\lambda = \frac{1}{SRA_d(a, a^*)},$$

where a^* is a list such that $a \cap a^* = \emptyset$.

Discounted Cumulative Gain : This formula outputs a value between 0 and 1, with 1 being given if both lists are identical, 0 when they have no elements in common, and values in between when there is partial overlap between elements and/or their order is different. $DCG_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i+1)}$

$$rel_i := \begin{cases} \frac{1}{rank_b(el_i)+1} & \text{if } el_i \in b \\ 0 & \text{otherwise} \end{cases}$$

Metrics Rejected

Kendall rank correlation [16]: This metric is bounded between 0 and 1 and compares the ranks of the elements of two lists. However, it cannot handle elements that only occur in one of the two lists, and thus is not suitable for our purposes. It also does not distinguish between differences in the upper and lower parts of the lists.

Spearman's footrule [35]: Rejected for the same reasons as Kendall rank correlation.

5.1.3 Sample text

ToDo: Show a sample paragraph with first n words from each list. Can be used to intuitively evaluate utilities In addition to the above automatic measures of list efficiency, we can also use human intuition to evaluate the various vocabulary lists: To compare vocabulary list up to some index k, we use a sample paragraph, and take the vocabulary $V_{l,k}$ of each of the lists. Then, we compare how the paragraph appears for each vocabulary if we blot out all words not in the vocabulary, in order to simulate the perspective of someone who only knows the words from the vocabulary.

5.2 Baselines

It may be useful to compare the lists generated by the various approaches with existing word lists from educational materials: Textbooks often feature chapters with word lists, or sentences which can be converted to word lists with a tokenizer. The purpose is to have a point of comparison, to see if generated lists agree with existing lists, and find reasons for differences.

5.2.1 Existing Lists

As one point of comparison, we can use existing vocabulary lists that are readily available in digital format, regardless of the method which created them. For this, a survey was performed on popular language learning tools which make public some of their learning data. However, mostly due to paywall restrictions, we only obtained one list for comparison, from the online learning platform *Rosetta Stone*(details below).

5.2.2 Textbooks

An obvious source of vocabulary lists to compare our lists against are English textbooks, which often contain vocabulary lists for each lesson. However, the books we considered as candidates do not publish their course contents free of charge. This includes:

- *Practical English Usage* by Michael Swan [2]

Cambridge Word Lists

Cambridge publishes word lists for certain levels of English [3]. However, these lists are meant for assessing learner's level of vocabulary skills, and the list for each level is ordered by alphabetical order, rather than the perceived level of the word, making them unusable as comparison to our approach.

Duolingo

While Duolingo is the most popular language learning application as of 2024, it does not publish its word lists or course contents that is free of cost and easily convertible to a format that can be processed with NLP tools.

Rosetta Stone

Rosetta Stone[4] is a popular online and mobile language learning platform. *Rosetta Stone* publishes some of the course contents on its website [5]. These lessons take the form of phrases that should help a learner to quickly gain competence in their target language. To make a list of vocabulary from *Rosetta Stone*'s lesson contents, we use the freely available lessons of the *American English* course (Units 1-20). We convert the PDF files to text format, filter lines which do not contain lessons contents (e.g., "Lesson 8" and similar ancillary notes). On the filtered text, we run the `bert-base-uncased` tokenizer and let the order of first appearance of a word in the lessons be its rank in the resulting vocabulary list. Thus, the generated list features the lessons' words in the order in which *Rosetta Stone* introduces them to its learners, which presumably is close to the word order that *Rosetta Stone*estimates to be most useful. One qualification of the previous statement is that some words may be introduced in order for the learner to have something concrete to talk about, rather than because of their estimated utility. The final vocabulary list contains 2,545 words in total.

This section has introduced some pre-existing lists as points of comparison for our generated vocabulary lists. However, one shortcoming of this comparison is that these lists have not been prepared with the same linguistic contexts in mind as those made by this work's approach, and thus their utility is naturally greater in general-context corpora than specialized ones. While the context specificity of this

[2]Available at Oxford University Press at `https://elt.oup.com/catalogue/items/global/grammar_vocabulary/practical_english_usage_4th_edition/9780194202510` (last accessed on April 25, 2025)

[3]`https://www.cambridgeenglish.org/images/149681-yle-flyers-word-list.pdf`

[4]`https://rosettastone.co.jp/`

[5]`https://support.rosettastone.com/s/article/Rosetta-Stone-Course-Contents`

work's utility extraction method is a major advance for creating vocabulary lists, it also makes this an unfair comparison.

However, apart from pre-existing lists, there also exist pre-existing **methods** of finding important words in texts, which we can also use as baselines to compare with our method. These also use corpora, but not AI models or XAI methods to find important words. The next section introduces some of these methods, which we use to generate comparison lists to our own.

5.2.3 Existing List Generation Methods

To get an impression of how well our XAI-based method for vocabulary list generation performs, we employ the three frequency-based methods for generating lists of vocabulary introduced in Section 2.5.2, namely Raw Frequency, Average Reduced Frequency, and TF-IDF. We use these three measures as baselines to our methods, using the same tokenizer `bert-base-uncased` to divide texts into words that we can count. To calculate the inverse document frequency in TF-IDF, we use a random sample of 53,178 documents from the Oscar corpus (see Section 4.3.2).

5.2.4 LLM Prompts

In recent years, Large Language Models have become a popular tools for language learners to find new words to learn about specific areas. For this purpose, we run the following prompt to ChatGPT 3 for each context to get a sense of how well our method performs against this easy method.

Prompt:

```
1   Given a learner of English who knows no English words so far:
2 Give me a list of 50 words from the script of the movie "Blade Runner"
    ↪ that they can learn, such that when reading the full script, they
    ↪  have the best understanding of what they are reading.
3 Order the list such that the most useful words appear at the top.
4 Do not group the words, just generate a raw list.
5 Do not number the bullet points. Do not put hyperlinks in the list.
```

Listing 5.1: Prompt given to the language model.

ToDo: results

5.3 Results

5.3.1 Correspondence of Efficiency Evaluation with Human Intuition

As mentioned in the introduction to this chapter, we use the performance of AI models on corpora where we simulate a limited vocabulary in order to measure the efficiency of a vocabulary list. Before we present the results of these tests, however, we examine whether the performance of AI models is a reliable indicator that the list is useful to humans as well. This section therefore examines several vocabulary

lists tested on a short sample text containing ten lines from subtitles of the 1972 movie *The Godfather*, seen here:

```
1  Moe loves the business.
2  He never said nothing about selling.
3  I'll make him an offer he can't refuse.
4  See, Johnny...
5  We figure that entertainment would draw gamblers to the casino.
6  We hope you'll sign a contract to appear five times a year.
7  Perhaps convince some of your friends in the movies to do the same.
8  We're counting on you.
9  Sure, Mike.
10 I'll do anything for my godfather.
```

Listing 5.2: Sample text.

When only the named entities and punctuation tokens are left in the text (our starting point for the evaluation), the text becomes: We display the evaluation score for each line on the left.

```
1  0.0                .
2  0.0                        .
3  0.0 '                    '       .
4  0.9  ,johnny...
5  0.0                                          .
6  0.0        '                                .
7  0.0                                               .
8  0.0  '             .
9  0.8   ,mike.
10 0.0 '                          .
```

Listing 5.3: Sample text with no vocabulary. Total score: 0.17.

We can see that the lines containing named entities (Line 4 and 9) already have a rather high score in the evaluation, reflecting the importance of the names to the meaning of the sentences. However, the other lines receive a score of 0.0, resulting in an overall score of 0.17. We also observe that the name *Moe* in line 1 is not recognized by the `spacy` named entity recognizer (falsely) making it a possible vocabulary word.

To compare their intuitive utilities, we use four vocabulary lists on the corpus and see how well their first words perform when evaluated on the sample text:

1. A randomly ordered list from words in the sample text.
2. A list compiled with raw frequency.
3. A list compiled with TF-IDF (with the Oscar corpus as normalization).
4. A list compiled by Single Token Summary with sentence embedding.

The tops of the lists can be seen in Table 5.1.

	Random	Raw Frequency	TF-IDF	Single Token Summary - Sent. Emb.
0	i	the	moe	moe
1	a	ll	godfather	casino
2	selling	we	gamblers	godfather
3	same	to	ll	gamblers
4	contract	he	convince	counting
5	make	i	counting	offer
6	draw	you	casino	refuse
7	of	a	refuse	ll
8	appear	do	loves	contract
9	casino	moe	draw	movies

Table 5.1: Top 10 words of each compared list on the sample text.

It can be seen that the ten top words of the lists collected with TF-IDF and the XAI Single Token Summary give a better idea of the meaning of the text than the words from the random and frequency lists. Next, we examine the words in the context of the sample text. In the following listings, we see the sample text with the 10 top words filled in from each vocabulary list, in addition to the recognized named entities. Each line displays the evaluation score for it, and the total score (calculated by taking the average across the lines) is seen in the caption of each listing.

```
1  0.0                .
2  0.2                    selling.
3  0.0 i' make              '       .
4  0.9  ,johnny...
5  0.7                          draw           casino.
6  0.3        '      a contract appear       a    .
7  0.1                of                            same.
8  0.0  '             .
9  0.8   ,mike.
10 0.1 i'                       .
```

Listing 5.4: Sample text with 10 words from the Random list. Total score: 0.3.

```
1  0.6 moe    the      .
2  0.1 he                      .
3  0.3 i' ll            he   '       .
4  0.9  ,johnny...
5  0.1 we                                  to the    .
6  0.1 we   you' ll   a       to              a    .
7  0.1                               the     to do the  .
```

```
0.3 we'          you.
0.8   ,mike.
0.2 i' ll do                .
```

Listing 5.5: Sample text with 10 words from the Raw Frequency list. Total score: 0.33.

```
0.8 moe loves      .
0.0                          .
0.5 ' ll                 '  refuse.
0.9  ,johnny...
0.7                           draw gamblers   casino.
0.1        ' ll                              .
0.3     convince                                    .
0.5  '   counting   .
0.8   ,mike.
0.6 ' ll             godfather.
```

Listing 5.6: Sample text with 10 words from the TF-IDF list. Total score: 0.52.

```
0.7 moe            .
0.0                          .
0.7 ' ll        offer    '  refuse.
0.9  ,johnny...
0.7                               gamblers    casino.
0.4        ' ll     contract                    .
0.4                                   movies          .
0.5  '   counting   .
0.8   ,mike.
0.6 ' ll             godfather.
```

Listing 5.7: Sample text with 10 words from the Single Token Summary list. Total score: 0.56.

The main observation is that the texts with the TF-IDF and Single Token Summary resemble the meaning of the original text much closer than the other two approaches, and are indeed evaluated significantly higher (0.52 and 0.56 vs. 0.33 and 0.30). In our opinion, this is evidence that our evaluation scores roughly align with how much the words let a reader understand the text, aligning with utility as defined in Section 2.1.2. Both of these lists feature the essential words *casino* and *godfather* and *gamblers*, as well as other words which let a reader guess at the situation.

We can, however, also notice some peculiarities in the evaluation scores: The text with words from the randomly assembled list (Listing 5.4) seems more understandable to use than when words from the frequency words are filled in (Listing 5.5), since the randomly compiled list contains useful words such as *selling*, *casino* and *contract*. However, the text with the frequency words receives a higher evaluation score than the randomly assembled list (0.33 vs. 0.3). This is due partially to the unrecognized entity *moe* which is contained in the frequency list but not the random one, "artificially" boosting the score of the frequency list. But even disregarding that name, the scores are not as disparate as we expected them to be. While this is necessarily a subjective observation, this seems to suggest that the evaluation scores do not reflect the true utility of words in all circumstances.

5.3.2 Small Context

This section shows the results of our evaluation approach for the efficiencies of vocabulary list, applied to lists which have been generated by the various list generation approaches. Performance scores were rounded to two decimals to improve readability.

Table 5.3 and 5.3 show the aggregated results of small-scale corpora, separated by corpus type. For small corpora, there is no train-test-data distinction, thus for each corpus and extraction method, a vocabulary list produced, whose efficiency is then evaluated with the same corpus as a basis.

	Raw Frequency	Average Reduced Frequency	TF-IDF	Attention - Sent. Emb.	Single Token Ablation - Sent. Emb.	Single Token Summary - Sent. Emb.	Progressive Summary - Sent. Emb.
0	you	you	i	you	you	you	you
1	i	i	you	i	i	i	i
2	the	the	me	the	what	we	it
3	to	to	t	s	it	it	what
4	s	s	yeah	it	no	no	no
5	a	a	okay	to	that	that	that
6	it	it	m	no	know	what	we
7	that	that	what	what	he	to	he
8	and	t	he	that	we	t	s
9	t	and	oh	a	me	here	me
10	of	of	s	is	to	he	to
11	we	is	re	and	this	go	the
12	is	what	gonna	we	s	know	know
13	what	in	don	yeah	the	yeah	is
14	in	we	my	t	is	don	not
15	me	me	it	this	right	okay	go
16	this	this	know	me	come	not	here
17	he	for	hey	of	not	she	this
18	your	on	we	okay	do	her	t
19	for	your	no	oh	go	oh	okay

Table 5.2: Tops of generated lists for OpenSubtitles dataset.

	Size=0	Size=10	Size=40	Size=160	Size=640
Raw Frequency	0.16	0.23	0.36	0.63	0.90
Average Reduced Frequency	0.16	0.23	0.36	0.62	0.90
TF-IDF	0.16	0.24	0.38	0.63	0.91
Attention - Sent. Emb.	0.16	0.24	0.38	0.64	0.90
Single Token Ablation - Sent. Emb.	0.16	0.26	0.40	0.67	0.95
Single Token Summary - Sent. Emb.	0.16	0.26	0.41	0.66	0.94
Progressive Summary - Sent. Emb.	0.16	0.25	0.40	0.66	0.94
[Mean]	0.16	0.24	0.38	0.64	0.92

Table 5.3: Model performance across vocabulary sizes on single subtitles.

	Size=0	Size=10	Size=40	Size=160	Size=640
Raw Frequency	0.51	0.57	0.65	0.77	0.91
Average Reduced Frequency	0.51	0.56	0.64	0.75	0.91
TF-IDF	0.51	0.61	0.68	0.78	0.92
Attention - Sent. Emb.	0.51	0.57	0.64	0.77	0.91
Single Token Ablation - NSP	0.51	0.57	0.64	0.76	0.90
Single Token Ablation - Sent. Emb.	0.51	0.61	0.69	0.82	0.96
Single Token Summary - Sent. Emb.	0.51	0.62	0.70	0.81	0.94
Progressive Summary - Sent. Emb.	0.51	0.61	0.68	0.80	0.95
[Mean]	0.51	0.59	0.67	0.78	0.93

Table 5.4: Model performance across vocabulary sizes on single Wikipedia articles.

	Raw Frequency	Average Reduced Frequency	TF-IDF	Attention - Sent. Emb.	Single Token Ablation - NSP	Single Token Ablation - Sent. Emb.	Single Token Summary - Sent. Emb.	Progressive Summary - Sent. Emb.	Random
Raw Frequency		0.99	0.64	0.94	0.98	0.74	0.68	0.65	0.52
Average Reduced Frequency	0.99		0.63	0.93	0.98	0.74	0.68	0.65	0.52
TF-IDF	0.64	0.63		0.62	0.54	0.80	0.83	0.78	0.44
Attention - Sent. Emb.	0.94	0.93	0.62		0.95	0.77	0.71	0.66	
Single Token Ablation - NSP	0.98	0.98	0.54	0.95		0.58	0.56	0.45	
Single Token Ablation - Sent. Emb.	0.74	0.74	0.80	0.77	0.58		0.90	0.85	0.43
Single Token Summary - Sent. Emb.	0.68	0.68	0.83	0.71	0.56	0.90		0.83	0.45
Progressive Summary - Sent. Emb.	0.65	0.65	0.78	0.66	0.45	0.85	0.83		0.45
Random	0.52	0.52	0.44			0.43	0.45	0.45	

Table 5.5: Similarity of vocabulary lists. Black boxes represent missing values.

5.3.3 Large Context

	Size=0	Size=10	Size=40	Size=160	Size=640	Size=2560
Raw Frequency	0.16	0.22	0.33	0.54	0.74	0.87
Average Reduced Frequency	0.16	0.22	0.33	0.54	0.73	0.87
TF-IDF	0.16	0.22	0.33	0.54	0.74	0.87
Attention - Sent. Emb.	0.16	0.23	0.34	0.55	0.74	0.87
Single Token Ablation - Sent. Emb.	0.16	0.23	0.34	0.55	0.74	0.87
Single Token Summary - Sent. Emb.	0.16	0.24	0.35	0.55	0.74	0.87
Progressive Summary - Sent. Emb.	0.16	0.23	0.35	0.55	0.74	0.87
[Mean]	0.16	0.23	0.34	0.55	0.74	0.87

Table 5.6: Model performance across vocabulary sizes on subtitles, with separate train/test subtitles.

	Size=0	Size=10	Size=40	Size=160	Size=640	Size=2560
Raw Frequency	0.53	0.58	0.62	0.70	0.80	0.89
Average Reduced Frequency	0.53	0.58	0.62	0.69	0.79	0.88
TF-IDF	0.53	0.60	0.63	0.70	0.79	0.88
Attention - Sent. Emb.	0.53	0.58	0.62	0.70	0.80	0.88
Single Token Ablation - NSP	0.53	0.58	0.62	0.69	0.79	0.88
Single Token Ablation - Sent. Emb.	0.53	0.59	0.63	0.70	0.79	0.88
Single Token Summary - Sent. Emb.	0.53	0.60	0.64	0.71	0.79	0.88
Progressive Summary - Sent. Emb.	0.53	0.59	0.62	0.68	0.78	0.87
[Mean]	0.53	0.59	0.62	0.70	0.79	0.88

Table 5.7: Model performance across vocabulary sizes on Wikipedia articles, with separate train/test articles.

	Raw Frequency	Average Reduced Frequency	TF-IDF	Attention - Sent. Emb.	Single Token Ablation - NSP	Single Token Ablation - Sent. Emb.	Single Token Summary - Sent. Emb.	Progressive Summary - Sent. Emb.
Raw Frequency		1.00	0.64	0.98	1.00	0.76	0.69	0.66
Average Reduced Frequency	1.00		0.64	0.98	1.00	0.76	0.69	0.65
TF-IDF	0.64	0.64		0.63	0.60	0.91	0.91	0.89
Attention - Sent. Emb.	0.98	0.98	0.63		0.98	0.75	0.68	0.63
Single Token Ablation - NSP	1.00	1.00	0.60	0.98		0.72	0.63	0.59
Single Token Ablation - Sent. Emb.	0.76	0.76	0.91	0.75	0.72		0.92	0.93
Single Token Summary - Sent. Emb.	0.69	0.69	0.91	0.68	0.63	0.92		0.88
Progressive Summary - Sent. Emb.	0.66	0.65	0.89	0.63	0.59	0.93	0.88	

Table 5.8: Similarity of vocabulary lists, big. Black boxes represent missing values.

We can observe several points in the data:

Single Token Ablation with sentence embedding achieves the highest results across corpus types. It is also among the approaches with the least spread of performance, as seen by the standard deviation of the corpora scores. Since the model used for list generation and list evaluation is the same, What we can gleam from this fact is that the utilities calculated from individual lines, when aggregated over the dataset, approach the word utilities with respect to the entire corpus better than frequency-based approaches.

The Progressive Summary approach, despite being a more complex approach, does not outperform Single Token Ablation. The comparison is difficult, as the calculation of line utilities for Progressive Summary is not as straightforward as for Single Token Ablation. However, Progressive Summary has the additional disadvantage of requiring a much greater amount of time to calculate for a corpus. For this reason, it seems that investigating possible improvements for how line utilities can be aggregated to calculate corpus utility may not be productive.

Single Token Summary produced the lowest performance across corpus types and vocabulary sizes, falling short of even the frequency-based baselines. This may be due to the fact that it does not take into account relationships between words in the sentence, as each summary contains only a single word. For this reason, Single Token Summary seems to be of very limited potential for word utility evaluation.

TF-IDF, despite its simplicity, produces lists of an efficiency very close to the XAI methods, especially at low vocabulary sizes. TF-IDF, while usually used to capture words characteristics of documents, also seems to capture useful words rather well. This could be seen as evidence to the effect that, while learning words by their general frequency provides the best coverage of texts, the coverage is not a good proxy for measuring the understanding of a text.

For the transformer attention approach, both the model for next sentence prediction and the model for sentence embedding produce very similar results. Because both models function with a BERT model at their core, this result is within the realm of expectation. The slightly better performance of the NSP model may be due to its greater size (12 attention heads vs. 6 heads), but the difference is too slight to make a definitive judgement.

We can also observe that the type of corpus used had a significant impact on the model performance for small vocabulary sizes: The average performance of 10- and 40-word-vocabularies was 0.37 and 0.53 for Wikipedia articles, but only 0.16 and 0.30 for subtitles, respectively. The discrepancy in performance of small vocabulary sizes between the two corpus types may be due to the fact that Wikipedia articles address a particular *topic*, meaning that knowing the main words (such as those from the article title) may produce a high initial boost in performance. One piece of counter-evidence to this theory is that raw frequency also performs better on Wikipedia than on OpenSubtitles, despite frequency lists from both corpus types featuring many stopwords at their top. An alternative hypothesis is that Wikipedia articles are more similar to the data the models had been trained on. This cannot be confirmed, however, as the source of BERT's training data is not publicly known. An interesting point to note is that, for a vocabulary size of 640, the list efficiencies

have nearly converged: Despite the larger average corpus size of movie subtitles, the average performance is 0.90 for subtitles and 0.89 for Wikipedia.

5.3.4 Performance Results of Lists Generated from Similar Corpora

So far, we have analyzed the efficiencies of vocabulary lists with respect to the same corpus which was used to generate them in the first place. This may be useful for a learner who wishes to read specific articles of watch specific movies, and learn the most useful vocabulary beforehand. However, in real life, it is not always possible for a learner to know the exact contents of the texts they will be engaging in: Natural conversations, for examples, are not deterministically predictable, and learners will likely wish to acquire abilities beyond the narrow scope of a few selected Wikipedia articles. For this reason, we also test the efficiencies of vocabulary lists on corpora which are similar, but not the same as the source of their generation.

To model these linguistic contexts, we use Wikipedia categories: For each category, we randomly split its articles into "train" and "test" articles, with a test percentage of 20%. When generating vocabulary lists, only the "train" articles are used as inputs. With these lists, we evaluate the list efficiencies against the "test" corpora. The scores across the individual corpora is then aggregated by taking the mean. Table 5.9 shows the results of these experiments.

							Performar
	Vocabulary Size	10	40	160	640	2560	102
Generation Method	Generation Model						
Raw Frequency	-	0.20	0.25	0.39	0.57	0.76	0.
Average Reduced Frequency	-	0.20	0.25	0.38	0.56	0.74	0.
TF-IDF	-	0.22	0.28	0.40	0.58	0.77	0.
Attention	nsp-bert	0.20	0.27	0.39	0.56	0.71	0.
	sent-transform_paraphrase-MiniLM-L6-v2	0.20	0.26	0.38	0.55	0.71	0.
Single Token Ablation	nsp-bert	0.20	0.26	0.39	0.55	0.72	0.
	sent-transform_paraphrase-MiniLM-L6-v2	0.21	0.24	0.39	0.55	0.72	0.
Single Token Summary	sent-transform_paraphrase-MiniLM-L6-v2	0.20	0.26	0.38	0.55	0.71	0.
Progressive Summary	sent-transform_paraphrase-MiniLM-L6-v2	0.19	0.25	0.37	0.53	0.70	0.
[Mean]	**-**	**0.20**	**0.26**	**0.39**	**0.56**	**0.73**	**0.**

Table 5.9: Performance of Vocabulary Lists on similar corpora.

Because vocabulary lists are generated from a much larger corpus, they are longer than the per-article lists in section 5.3.2. We can also observe that the increase in performance is much slower than when a list is evaluated on the same corpus used to generate it.

5.3.5 Speed

When evaluating the broad applicability of a vocabulary list generation approach, the efficiency of generated vocabulary lists is not the only criterion: The runtime of the approach is also an important factor. Table 5.10 shows the average runtime for the approaches per corpus type. It must be kept in mind that the runtime is not

only influenced by the word utility evaluation method, but also by implementation details, such as whether batch processing of inputs by the AI models is exploited, and the size of the model employed. Named Entity Recognition is performed within each method, and contributes most of the runtime for the frequency-based approaches.

		Runtime [s]	
	Corpus Type	OpenSubs Subtitle	Wikipedia Article
Generation Method	Generation Model		
Raw Frequency	-	5.4	2.1
Average Reduced Frequency	-	5.4	2.1
TF-IDF	-	6.2	2.9
Single Token Ablation	NSP	-	115.6
	Sent. Emb.	54.5	48.3
Single Token Summary	Sent. Emb.	53.2	26.1
Progressive Summary	Sent. Emb.	187.0	256.1

Table 5.10: Runtime of list generation approaches.

As is to be expected, the frequency-based approaches are orders of magnitude faster than the XAI-based approaches. Among the XAI approaches, we can make several observations: First, there is a marked difference in runtime between the NSP model and the sentence embedding model. This seems to be due both to the greater complexity of the NSP model, and the fact that it processes more data in one call (a pair of lines instead of a single line). Attention is the fastest list generation approach, owing to the fact that only one model call is necessary per line explanation. Attention could be made even faster with batch processing, but since it is a fairly efficient method already, we did not optimize it to the fullest extent.

On average, most of the XAI approaches take more time on subtitles than one Wikipedia articles, with the exceptions of Progressive Summary. This aligns with our understanding of the corpora and the approaches 4.3: Wikipedia Articles have fewer lines, but more words per line, making the Progressive Summary approach take longer due to its time complexity of $\mathcal{O}(n^2)$.

5.4 Discussion

ToDo: Interpretation, Could you combine approaches with each other or baselines to combine strength, what are the strengths/weakness of each approach

Chapter 6

Outlook

We have seen in the previous chapter that our approach to list generation outperforms frequency-based metrics when ToDo: insert exact circumstances. This final chapter discusses several possible improvements to our approach and the research around it, which were not possible within the scope of this work due to time constraints.

6.1 Lemmatization

The vocabulary lists we compiled were dictated by the tokens provided by the tokenizer. With this approach, each unique combination of letters is regarded as an independent word, including variations of a particular word such as "house" and "houses", "look", "looks", "looked". However, in textbooks, vocabulary lists are usually organized by headwords or *lemmas*, that is, the variations as seen above are considered as a single word. To compile more useful vocabulary lists, we recommend that words which derive from the same lemma be treated as the same word. "Treating" here includes the masking of tokens for model-agnostic XAI methods, as well grouping words together by their lemma in the post-processing of vocabulary lists which are produced by model-specific approaches.

6.2 Active vs. Passive Utility

This work has exclusively focused on the utility of words for understanding existing texts, however, one can also consider the utility of a word for active speaking and writing: One obvious difference is that, when there are two synonyms, it suffices to know one of the two for active speaking, but for passive understanding, both must be learned.

6.3 Multilinguality

While we have made an effort to make our implementation handle as many languages as possible, we have only tested it on English corpora. An extension of this work should include evaluation on the other languages supported by the data pipeline.

6.4 Distinction of Homonyms

Some words can have multiple meanings: An example of this is the word *can*, which can signify either a container (a tin can) or the modal verb (to be able to do something). Such words are called *homonyms*. In order to produce more useful lists of vocabulary, distinction between the different meaning of such words could be a useful component, especially when one of the meanings is more common than the other.

Chapter 7

Appendix

7.1 Abbreviations

NSP Next sentence prediction

NLP Natural language processing

LLM Large language model

Bibliography

[1] Alexander Clark, Chris Fox, and Shalom Lappin. The Handbook of Computational Linguistics and Natural Language Processing: Introduction. In *The Handbook of Computational Linguistics and Natural Language Processing*, pages 1–8. John Wiley & Sons, Ltd, 2010.

[2] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. Text Summarization Techniques: A Brief Survey, July 2017.

[3] Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the association for computational linguistics*, 7:597–610, 2019.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Noam Dahan and Gabriel Stanovsky. The State and Fate of Summarization Datasets: A Survey, February 2025.

[6] Claus Thorn Ekstrøm, Thomas Alexander Gerds, Andreas Kryger Jensen, and Kasper Brink-Jensen. Sequential rank agreement methods for comparison of ranked lists, August 2015.

[7] Matthieu Futeral, Armel Zebaze, Pedro Ortiz Suarez, Julien Abadji, Rémi Lacroix, Cordelia Schmid, Rachel Bawden, and Benoît Sagot. mOSCAR: A Large-scale Multilingual and Multimodal Document-level Corpus, June 2024.

[8] Razieh Gholaminejad and Mohammad Reza Anani Sarab. Academic vocabulary and collocations used in language teaching and applied linguistics textbooks: A corpus-based approach. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 26(1):82–107, June 2020.

[9] Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*, volume 29, pages 31–43, 2012.

[10] Xuehong (Stella) He and Aline Godfroid. Choosing Words to Teach: A Novel Method for Vocabulary Selection and Its Practical Application. *TESOL Quarterly*, 53(2):348–371, June 2019.

[11] S. Hunston. Corpus linguistics. In Keith Brown, editor, *Encyclopedia of Language & Linguistics (Second Edition)*, pages 234–248. Elsevier, Oxford, second edition edition, 2006.

[12] Sarthak Jain and Byron C. Wallace. Attention is not Explanation, May 2019.

[13] Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. The State and Fate of Linguistic Diversity and Inclusion in the NLP World, January 2021.

[14] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. (unpublished), 3rd edition, 2025.

[15] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. (unpublished), 3rd edition, 2025.

[16] M. G. KENDALL. A NEW MEASURE OF RANK CORRELATION. *Biometrika*, 30(1-2):81–93, June 1938.

[17] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[18] Sofie Johansson Kokkinakis and Elena Volodina. Corpus-based approaches for the creation of a frequency based vocabulary list in the EU project KELLY–issues on reliability, validity and coverage. *Proceedings of eLex*, 2011:129–139, 2011.

[19] Tao Lei. *Interpretable Neural Models for Natural Language Processing*. Thesis, Massachusetts Institute of Technology, 2017.

[20] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding Neural Networks through Representation Erasure, January 2017.

[21] Shaofeng Li, Phil Hiver, and Mostafa Papi. *The Routledge Handbook of Second Language Acquisition and Individual Differences*. Routledge, April 2022.

[22] Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016.

[23] Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions, November 2017.

[24] Michael West. *A General Service List of English Words*. Longman, Green and Co., London, 1953.

[25] Christoph Molnar. Chapter 4: Methods. In *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. (unpublished), 3 edition, 2025.

[26] P. Nation. Vocabulary size, text coverage and word lists. *Vocabulary: Description, acquisition and pedagogy/acquisition and pedagogy*, pages 6–19, 1997.

[27] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky,

Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, March 2024.

[28] Shahzad Qaiser and Ramsha Ali. Text mining: Use of TF-IDF to examine the relevance of words to documents. *International Journal of Computer Applications*, 181(1):25–29, 2018.

[29] Dragomir Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Computational linguistics*, 28(4):399–408, 2002.

[30] Nils Reimers and Iryna Gurevych. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation, October 2020.

[31] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, August 2016.

[32] E. Rich. *Artificial Intelligence*. Artificial Intelligence. McGraw-Hill, 1983.

[33] Petr Savický and Jaroslava Hlavácová. Measures of Word Commonness. *Journal of Quantitative Linguistics*, 9(3):215–231, December 2002.

[34] John R. Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(3):417–424, 1980.

[35] Charles Spearman. Correlation calculated from faulty data. *British journal of psychology*, 3(3):271, 1910.

[36] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT–building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, 2020.

[37] Lindia Tjuatja, Valerie Chen, Tongshuang Wu, Ameet Talwalkwar, and Graham Neubig. Do LLMs Exhibit Human-like Response Biases? A Case Study in Survey Design. *Transactions of the Association for Computational Linguistics*, 12:1011–1026, 2024.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. *Advances in neural information processing systems*, 30, 2017.

[39] Giulia Vilone and Luca Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106, 2021.

[40] Sarah Wiegreffe and Yuval Pinter. Attention is not not Explanation, September 2019.

[41] Ismail Xodabande, Mahmood Reza Atai, Mohammad R. Hashemi, and Paul Thompson. Developing and validating a mid-frequency word list for chemistry: A corpus-based approach using big data. *Asian-Pacific Journal of Second and Foreign Language Education*, 8(1):32, October 2023.