ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

School of Engineering
Master Degree in Artificial Intelligence

READABILITY ASSESSMENT FOR TEXT SIMPLIFICATION

Thesis in
Machine Learning

Supervisor
Chiar.mo Prof.Ing. Claudio Sartori

Co-supervisor
Mrs Anna Elisabetta Ziri

Candidate
Reza Shatery

Second Session
Academic Year 2024

Keywords

Text Simplification
Readability Assessment
Text processing
Feature extraction
Simplification evaluation

dedica

1) Introduction

Text simplification involves altering a text to make it easier to read or understand without changing its original meaning. The objective is to create content that is more accessible for specific users or systems. Initial studies, such as those by Chandrasekar et al. (1996)[1], focused on enhancing parsing efficiency by breaking down lengthy sentences into shorter ones. In the field of Natural Language Processing (NLP) and computational linguistics, the simplification of sentences is an essential task aimed at converting complex sentences into simpler, more comprehensible forms without altering their original meaning. This process is critical for enhancing the accessibility and readability of text for diverse audiences, including children, language learners, and individuals with cognitive impairments.

Complex sentences, characterized by their intricate structures, multiple clauses, and frequent use of subordinate clauses, can often obscure the primary message, making comprehension challenging. Simplification addresses these challenges by breaking down complex sentences into simpler, more manageable components. Simplified sentences are easier to read and understand, which is crucial for audiences with varying levels of language proficiency. Clearer and less ambiguous sentences allow NLP systems to extract information more accurately. Simple sentences provide better input for machine translation and summarization systems, leading to more effective and accurate outputs. Additionally, simplified text is more accessible to individuals with cognitive disabilities or reading difficulties, promoting.

To achieve effective sentence simplification, it is necessary to identify and extract various linguistic features. These features aid in understanding the structure and content of sentences, thereby enabling accurate and meaningful simplification. Key linguistic features include syntactic structure, which involves the analysis of the grammatical structure to identify main and subordinate clauses, conjunctions, and dependencies. Lexical complexity assesses the difficulty of vocabulary, including the identification of rare or complex words that can be replaced with simpler alternatives. Sentence length is measured, as longer sentences often require splitting into shorter ones. Clause analysis involves the identification and differentiation of independent and dependent clauses to simplify or separate them appropriately. Understanding the semantic roles of different words and phrases within a sentence helps maintain meaning while simplifying the structure. By leveraging these linguistic features, NLP systems can transform complex sentences into simpler ones while preserving essential information and meaning. This process not only benefits end-users by improving text accessibility and comprehensibility but also enhances the efficiency and accuracy of various NLP applications.

this is the general workflow of this research.

- **Introduction**
  - Review of existing research and methodologies in Natural Language Processing (NLP) and text simplification.
  - Introduction to readability assessment and the need for simplified text for diverse audiences.
  - Key objectives and motivations for the study.
- **Data Preparation Techniques**
  - **Word Segmentation & Tokenization**: Preparation of datasets by breaking text into tokens.
  - **Normalization of Words & Lemmatization**: Reduction of words to their base forms.
  - **Parsing & Syntactic Dependencies**: Extraction of structural relationships between words in a sentence.
  - **Libraries Used for Implementation**: Tools like SpaCy and TreeTagger for linguistic feature extraction.
- **Feature Exploration**
  - **Types of Features**: Lexical, syntactic, morphological, and semantic features used to distinguish between simple and complex sentences.
  - **Feature Extraction for Readability**: How these features help assess sentence complexity and readability.
  - **Feature Selection Methods**: Use of techniques like Recursive Feature Elimination (RFE) for selecting the most relevant features.
- **Datasets**
  - **English Dataset**: Usage of crowdsourced datasets for sentence complexity evaluation in English.

  - **Italian Dataset**: Adaptation of the features and classifiers to test their effectiveness in another language.
  - **New Dataset Using GPT-4o**: Creation and evaluation of a new dataset using GPT-4o to classify simple and complex sentences.

- **Feature Selection and Implementation**
  - **Recursive Feature Elimination (RFE)**: Use of RFE for selecting the best features in combination with classifiers.
  - **Biserial Correlation Coefficient**: Evaluation of feature relevance using statistical measures like the biserial correlation coefficient.
  - **Model Training and Cross-Validation**: Training models using cross-validation techniques to optimize accuracy and performance.

- **Conclusion and Future Work**
  - Summarizes the results and effectiveness of models, feature selection techniques, and performance across different datasets.
  - Highlights the key takeaways of the research, emphasizing contributions to sentence classification tasks.
  - Suggests potential expansions of the research, including applying the models to other languages and datasets, investigating advanced techniques, and implementing real-time classification systems.

2) Data Preparation techniques

A common practice in NLP is text normalization: a set of steps to transform a text into its canonical form [Jurafsky and Martin][2].

2.1 Word segmentation: tokenization

A first step is tokenization: the segmentation of the elements that make up the text into different tokens. For example, the sentence "The sun rises, the day begins." is divided into the tokens "The," " sun," "rises," "," and so on. Usually words are divided by spaces or punctuation, but it is not always enough to consider these delimiters for division: for example, the word "rock'n'roll," as well as "New York," should be considered one token, while "the element" should be divided into two.

2.2 Normalization of words: lemmatization

Another important step is that of lemmatization, or the reduction of words (or rather, tokens) to their basic form: the lemma. A token can, in fact, appear in morphologically different forms. For example, the verb "to be" can be written in a variety of forms, such as "was," "is," "was," "will be," and so on, while being traced back to a single entry in the dictionary. Interestingly, the number of distinct lemmas in a text is much smaller than the number of tokens and is governed by Heaps' empirical law: $|V| = kNB$ where $|V|$ is the number of distinct lemmas, N the number of tokens in a text and k and B two language-dependent parameters [Heaps 1978][3] (for English k € [10, 100], B € [0.4, 0.6]). Reducing all tokens to their lemma reduces the total number of tokens, simplifying subsequent analysis. Lemmatization is a complex process, as it requires disambiguating the word: for example, distinguishing whether it is used as an adjective or noun by evaluating its context. Distinguishing and labeling the lexical category (or tag) of the token is called part-of-speech (POS) tagging. There are several approaches to perform this operation. The implementation used in TreeTagger, based on decision trees, will be described.

2.3 Parsing: analysis of syntactic dependencies

Elements of linguistics

To discuss how a parser works and how information about syntactic dependencies between words in a sentence can be derived, it is necessary to intro-durate some elements of linguistics [Berruto and Cerruti 2011][4]. In linguistic analysis, a sentence is defined as an information unit that contains a proposition. A sentence is decomposable into immediate constituents, generally labeled syn- tagms : combinations of two or more words constituting a syntactic unit. Not all sets of words form a syntagma; to be defined as such, they must meet certain criteria including mobility, i.e., the set of words must be considered as a movable block within the sentence, and isolability, the ability of the set to form an utterance on its own. There are several categories of

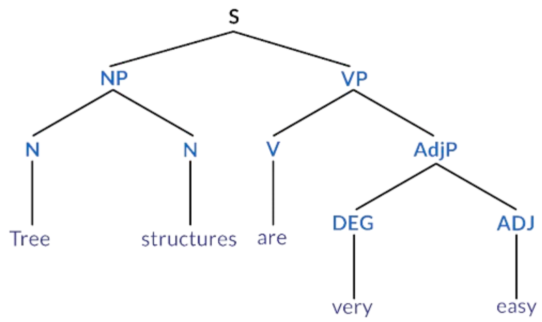

Figure 1: Example of the syntactic tree of the sentence

syntagms, named according to their "head": the main element of the word set. Some examples are the nominal, verbal and adjectival syntagma.The most frequently used method for representing sentences is syntactic trees, in which each node consists of two branches and the leaves represent the words that make up the sentence. For example, Figure 1 shows the syntactic tree of the sentence "Tree structures are very easy." The sentence, represented by the letter S, is decomposed into two syntagms: a noun phrase (NP), consisting of a name. (Tree) and a noun (N) and a verbal phrase (VP) consisting of the verb (V) "are" and the Adjective phrase consist of degree (very) and adjective (easy).

2.4 Formal grammar and parsing

For dependency analysis (parsing) there are three main approaches: the transition-based approach, in which the syntactic tree is built incrementally, first defined in [5], the graph-based approach, which treats the tree as a global optimization problem[6], and the probabilistic approach. SpaCy, the library used in this study that will be discussed later, implements parsing with a particular implementation [?] of the transition-based approach. In order to discuss how the original approach works, it is necessary to intro-durate the concept of formal grammar, particularly context-free grammar. A formal grammar is defined as an abstract structure that describes a language by means of a set of precise rules. In[7] Noam Chomsky forma- lized a set of 4 formal grammars, organized in a hierarchical structure, described by a set of rules and productions that govern how each constituent of a sentence can be transformed. Among them, the context-free grammar can be used to model the structure of the constituents of a natural language. Formally, a context-free grammar is defined by quadruple of parameters (VN,VT,P,F) in which:

VN: set of non-terminal symbols
VT: set of terminal symbols (disjoint from VN)
P: set of production rules:
  each in the form A $\rightarrow$ B
  in which A is nonterminal
  B is a string of symbols from the set $V_T \cup V_N$
F: the initial and member symbol of V_N

The language produced by the formal grammar is generated by following the productions of the set P, so from one symbol another symbol can be derived by applying the production rule. Returning to the transition-based parser, this takes its origin from the Shift-Reduce parsing technique originally developed to parse programming languages [8]. This type of parser consists of three elements: a stack in which the tree is built, a buffer containing the tokens to be processed, and the parser that parses them. Parsing is done by a prediction made by another component of the parser: the oracle. Scrolling through the buffer, the parser parses the sentence from left to right token by token, consulting the oracle to know which operation to perform. Looking at the two elements at the top of the stack and the token in the buffer, the oracle can decide to reduce the stack by removing one element (Reduce) or to remove the token from the buffer to insert it into the stack (Shift). The parser analysis thus operates bottom-up, starting with the individual tokens in the

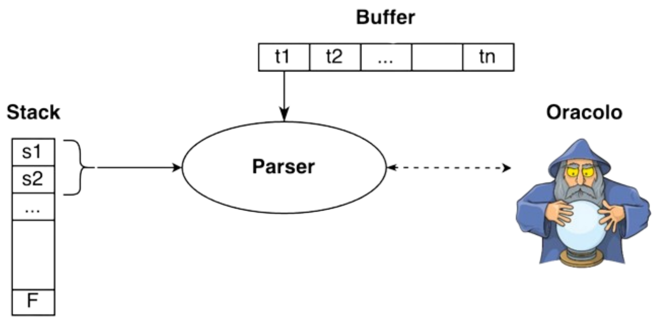

Figure 2: Diagram of the parser

sentence to be analyzed and tries to reduce it to the initial symbol F. Specifically, the set of possible operations is: LeftArc: assigning a relationship between the token at the top of the stack and the next token on the stack and removing it from the stack (Reduce) RightArc: assigning a relationship between the second token from the top of the stack and the first one and removing the latter from the stack (Reduce) Shift: token removal from buffer and insertion at top of stack. This set of operations, defined in[5], is called arc-standard. Variations of it have been proposed over the years to improve its operation, for example, the arc-eager approach modifies the LeftArc and RightArc operations, causing them to assign relationships between the top of the stack and the token in the buffer, rather than between the two elements at the top of the stack.

In addition, the RightArc operation moves the token from the buffer to the stack instead of removing the top of the stack, and a new action (Reduce) is added to the assembly that allows the removal of the top of the stack. This variation provides greater flexibility in the assignment of dependencies, allowing a reduction in error. To select the right operation, the oracle follows the rules of the considered grammar, assigning a relation when it encounters a production rule of the set P. The parser, starting from a stack containing only the start symbol F and the first token, proceeds to parse the entire buffer, forming the dependency tree in the stack. Having to produce a syntactic tree, the assignment of a relation can be parameterized by a label that determines its syntactic relation.

The creation of the oracle for natural language parsing is generally done by machine learning methods. There are a variety of implementations of these, based on manual feature extraction, using, for example, SVM classifiers [9], or neural classifiers [10] The parser implemented in the spaCy library[11], which was used for analysis in this study, combines the ideas of two previous works achieving an improvement in performance. The first work is [9], in which the Unshift operation is added to the parser, which allows the token to be moved to the top of the stack in the buffer if the latter is empty and the token does not have a relationship as a "head" with respect to another token. The second is [12] in which an approach is proposed that overcomes some limitations of the arc-eager approach.

2.5 Libraries used for implementation

spaCy

SpaCy is an open-source library equipped with several high-level NLP features, including Named-Entity Recognition (NER), part-of-speech (POS) labelling, syntactic dependency analysis, text segmentation (tokenization), and lemmatization. These operations are carried out through a pipeline that allows a series of processing steps to be performed in sequence, each adding a layer of annotation to the input document. One of the most distinctive aspects of spaCy is its object-oriented data model: each input text is processed by the tokenizer creating the Doc object, which is subsequently processed by the pipeline, which processes its features. The library supports many languages with ad-hoc pipelines. For the english language, 3 different pipelines are available, distinguished by their dimention and the processing carried out.

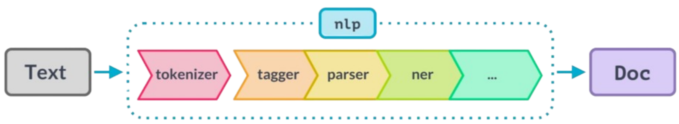

Figure 3: the pipline of spacy

the pipeline consists of 6 different components:

- Tok2Vec: the tokenizer.
- Morphologizer: predicts morphological features of the token, such as whether it is in the plural or singular form, gender, verbal mode etc.
- Tagger: assigns a tag to each token, based on its category, distinguishing adjectives, verbs, pronouns, nouns and other lexical categories.
- Parser: parses the syntactic dependencies of the text, making syntactic tree navigation available.
- Lemmatizer: assigns each token its basic form using rules based on its tag.

- Named-Entity Recognition (NER): a component capable of identifying tokens that correspond to predefined categories, such as personal names, places, organization names, dates and times, allowing detailed classification.

2.6 Implementation choices

An additional feature of spaCy is the ability to customize the pipeline, allowing custom components to be inserted and existing ones modified. That this possibility was used to replace the default Lemmatizer with TreeTagger, shown below, which demonstrates excellent results in lemmatization. In addition, a component was added at the beginning of the pipeline to better partition the input text. Reading text from documents, in fact, it is usual to find bulleted lists and sentences without a period (e.g., titles) that are difficult to interpret. correctly as different sentences from spaCy. The added component takes care of facilitating sentence segmentation by inserting punctuation. In addition, it was chosen to discard all sentences for which at least one proposition (or clause) is not identified, attempt to ignore headings and notes within the document. In this way, sentences that are too simple-nominal-are not considered as actual parts of the text, thus excluding them from the calculation of other readability indices, allowing a more concrete analysis of the document.

2.7 TreeTagger

TreeTagger is a tool developed by Helmut Schmid, University of Stuttgart, for POS and text lemmatization for many languages. Presented in [13] and [14], it introduces a method of tagging parts of speech using decision trees, improving on results obtained by previous probabilistic algorithms. Previous to the study, in fact, n-grams were used to determinate the lexical category of the word through a probabilistic calculation [15], [16], called n-gram Taggers. TreeTagger, as well as the n-gram Taggers, models the lexical category (or tag) of the word using the Markov Model, whereby the tag of the current word is determined based on the tags of the words that precede it. Using a bigram the previous tag is observed, with a trigram up to two previous tags, and so on. For example, using a trigram.

$$p(w\,w_{12}\ldots w_n,\, t\,t_{12}\ldots t_n) := p(t_n \mid t\,t_{n-2n-1})p(w_n \mid t_n)p(w\,w_{12}\ldots w_{n-1},\, t\,t_{12}\ldots t_{n-1})$$

Where w are the words and t corresponding tags. The term $p(w_i \mid t_i)$ denotes the probability of the word wi given the tag t_i, while $p(t_i \mid t_{i-2}\,t_{i-1})$. denotes the probability that the tag ti will occur after the previous two tags $t_{i-2}\,t_{i-1}$.

In n-gram taggers, the probability of the tag is estimated by observations (Maximum likelihood estimation).

$$p(t \mid t_{n\,n-2\,n-1}) = \frac{F(t_{n-2}t_{n-1}t_n)}{F(t_{n-2}t_{n-1})}$$

Where $F(t_{n-2}t_{n-1}t_n)$ and $F(t_{n-2}t_{n-1})$ are respectively the number of occurrences of the trigram $t_{n-2}t_{n-1}$. tn and the bigram $t_{n-2}t_{n-1}$. in the corpus. In words, an attempt is made to determine the probability of a certain tag given the previous tags by observing how many times the occurrence of those tags is followed by the tag in question. However, this type of approach is problematic in analyzing uncommon trigrams, having a low number of observations. It also requires a very large corpus to determine reliable probability estimates for each n-gram. Tree Tagger, on the other hand, determines the word tag using binary decision trees. The probability of a trigram is calculated by following the path from the root of the tree to a leaf, where at each node of the tree the path is divided by the value of a previous tag and the final leaf contains the probability of the last tag. The problem of defining the tag of a word is seen as a classification problem, in which the previous tags are the features. The number of tags taken into consideration can be decided when constructing the decision tree, and an increase in them improves performance, however, requiring a larger corpus. The advantage of TreeTagger's approach is to achieve high accuracy (ca. 97%) without requiring very large corpus.

2.8 Analysis of extracted information

Using the tools discussed, the case study of the project, was analysed in order to analyse their text features and how they were related to each other. Characteristics extracted from the text are:

- Sentence length: number of words contained.
- Average word length: number of characters related to the number of words.
- Percentage of Uncommon Words (Out of Vocabulary (OOV): calculated as the percentage of words (reduced to lemma) not belonging to the vocabulary, not calculating words recognized as entity names by the NER component.
- Percentage of subordinates: number of subordinates per main, per sentence.

- Syntactic tree depth: calculated as the maximum depth of the syntactic tree. Length of dependency relations: average length of dependencies in the syntactic tree.

3) Feature exploration

3.1 Types of Features

In previous study for Italian language, they used 7 features for check whether a sentence is simple or complex.the features are: Characters, Average length of dependencies, words not in the dictionary, max of depth tree, tokens, subordinate. With this features for English language, the accuracy for SVM classifier was: 84% and f1-score is 76. The previous study just used 7 Features and to have better accuracy more features are needed , so in this project more features are extracted for English Language.

Traditional Features: Traditional Features refer to straightforward computations of the surface-level properties of a text. The evaluated metrics include the text's sentence count, the average and maximum words per sentence, the average characters per word, and the average syllables per word. Additionally, two widely-used readability formulas are incorporated: the Flesch-Kincaid score [17] and the Coleman-Liau readability index [18].

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

But Flesch-Kincaid score should not be the only metric for the readability assessment[19], because this score only care about the number of words and syllables. based on [20] there are categories for features that are more important. These features collectively cover a broad range of linguistic aspects, including lexical richness, syntactic structure, morphological complexity, and readability, which are crucial for determining sentence complexity. Here's why these features are useful:

- **Lexical Features:** are characteristics or properties of words used in natural language processing (NLP) to analyze and understand text. These features capture various aspects of words, such as their identity, form, function, meaning, and usage patterns.

- **Syntactic Features:** are characteristics or properties of the structure and arrangement of words within sentences used in natural language processing (NLP) to analyze and understand text. These features capture various aspects of syntax, such as the grammatical relationships between words, sentence structure, and the roles that words play within sentences.
- **Morphological Features:** are characteristics related to the form and structure of words in natural language processing (NLP). These features capture aspects of word formation and modification, such as inflection, derivation, and composition.
- **Semantic Features:** are characteristics related to the meaning of words and phrases in natural language processing (NLP). These features capture various aspects of semantics, including the relationships between words, their meanings, and their contextual usage.

For the English language, 23 features are selected based on their significance. Since each feature has an index and there is no established criterion to determine which feature is superior, the features are initially implemented specifically for the English language.

3.2 features for assessing simple and complex sentences

- **Average syllables:** Average number of syllables per word for each sentence entry in the dataset.
- **Incidence of relative clauses:** The "incidence of relative clauses" refers to the frequency or occurrence of relative clauses in a body of text or language. Relative clauses are subordinate clauses that modify a noun phrase and provide additional information about it. They typically begin with relative pronouns such as "who," "which," "that," or "whom." For example, in the sentence "The book that I bought is on the table," the relative clause "that I bought" modifies the noun "book" and provides information about which book is being referred to[21].

- **Incidence of apposition:** The "incidence of apposition" refers to the frequency or occurrence of apposition structures in a body of text or language. Apposition is a grammatical construction in which two elements, typically noun phrases, are placed next to each other, with one element providing additional information or clarification about the other. For example, in the phrase "my friend John," the noun phrase "my friend" is in apposition to the noun "John," providing additional information about who John is. The incidence of apposition can vary depending on factors such as the style of writing, the complexity of the language, and the specific context in which the language is used. Analysing the incidence of apposition structures can provide insights into the structure and complexity of a language or text, as well as the writer's stylistic choices.[22].

- **The number of arity of verbal predicates:** The arity of verbal predicates refers to the number of arguments or participants that a verb takes in a sentence. It describes the number of noun phrases or other syntactic elements that are required or allowed to accompany the verb to form a grammatically complete sentence[23].

- **Number of negative causal connectives:** The "Number of negative causal connectives" refers to the count of negative causal conjunctions or connectives in each text or context. Negative causal connectives are words or phrases that are used to express a causal relationship between two clauses or ideas, but with a negative implication or outcome. They typically indicate that one event or situation causes or results in another event or situation, but with a negative consequence or effect. Examples of negative causal connectives include: Because of + negative outcome, Due to + negative outcome, As a result of + negative outcome, Owing to + negative outcome[24].

- **Number of positive additive connectives:** The "Number of positive additive connectives" refers to the count of positive additive conjunctions or connectives in each text or context. Positive additive connectives are words or phrases that are used to add similar or related information, typically indicating addition, inclusion, or amplification of a preceding idea or clause. Examples of positive additive connectives include: "and", "also", "as well as", "furthermore"[24].

- **Embedded complement 'chains':** Embedded complement "chains" refer to a syntactic structure in which a verb takes an embedded complement clause, and within that complement clause, there are additional embedded complement clauses. This creates a hierarchical or nested structure of complement clauses within one another. For example:
She said [that [he knew [that [they were coming]]]]."[25]

- **Length of the verbal root:** In readability assessment, the "Verbal Root" refers to the main verb or the root verb of a sentence. Identifying the verbal root is important as it often provides key information about the action or main idea expressed in the sentence. In many cases, readability formulas or assessments use features related to the verbal root to evaluate the complexity of a sentence. For example, the length or complexity of the verbal root, the presence of auxiliary verbs, or the syntactic structure around the verbal root may all contribute to determining the readability or complexity of a sentence.[26]

- **Main verb – inf. mood:** "Main verb – inf. mood" refers to the main verb in a sentence that is in the infinitive mood. The infinitive mood is a verb form that expresses an action or state without indicating the subject or tense. In English, infinitive verbs are typically preceded by the word "to". For example:
"I want to go to the store." (The main verb "want" is in the indicative mood, and the infinitive verb "to go" is the main verb in the infinitive mood.) "She decided to study abroad." (The main verb "decided" is in the indicative mood, and the infinitive verb "to study" is the main verb in the infinitive mood.) Identifying main verbs in the infinitive mood can be important for understanding the structure and meaning of sentences, as infinitive verbs often serve as the main actions or intentions expressed by the subject.[27]

- **Number of nouns:** The number of nouns in each sentence of a dataset.

- **Aux.verb – indic.mood:** Refers to an auxiliary verb that is in the indicative mood. Let's break down the components: Auxiliary Verb (Aux. verb): An auxiliary verb, also known as a helping verb, is a verb that accompanies the main verb in a clause to convey grammatical information such as tense, aspect, mood, and voice. Examples of auxiliary verbs in English include "be," "have," and "do." Indicative Mood (indic. mood): The indicative mood is a verb form used to make factual statements or ask questions about real or apparent states of affairs. It is the most common mood in English and is used to express certainty or reality. When a verb is in the indicative mood, it indicates that the speaker is making a statement or asking a question about something that is believed to be true or is happening in reality. Putting them together, "Aux. verb – indic. mood" refers to an auxiliary verb that is functioning in the indicative mood, indicating that it is being used to convey factual information or statements about real or apparent states of affairs in a clause[28].

- **Modal verb – indic.mood:** Refers to a modal verb that is used in the indicative mood. Modal Verb: Modal verbs are a type of auxiliary verb that express modality, indicating possibility, necessity, permission, ability, or obligation. Examples of modal verbs in English include "can," "could," "may," "might," "must," "shall," "should," "will," and "would." Indicative Mood: The indicative mood is a verb form used to make factual statements or ask questions about real or apparent states of affairs. It is the most common mood in English and is used to express certainty or reality. When a verb is in the indicative mood, it indicates that the speaker is making a statement or asking a question about something that is believed to be true or is happening in reality. When combined, "Modal verb – indic. mood" refers to a modal verb that is functioning in the indicative mood, indicating that it is being used to convey factual information or statements about real or apparent states of affairs in a clause. Modal verbs in the indicative mood often express possibilities, permissions, or likelihoods in a straightforward manner.[28]

- **Number of Modifier:** A "modifier" is a grammatical term used to describe a word or phrase that provides additional information about another word in a sentence. Modifiers can be adjectives, adverbs, phrases, or clauses that add details, descriptions, or qualifications to nouns, pronouns, verbs, or other modifiers.In the context of natural language processing (NLP) and linguistic analysis, a "modifier" refers to any word or phrase that serves to modify or describe another element in the sentence. For example:
  In the phrase "the tall tree," the word "tall" is a modifier that provides additional information about the noun "tree." In the sentence "She quickly ran to the store," the adverb "quickly" is a modifier that describes how the action of running was performed. In the phrase "the book on the table," the prepositional phrase "on the table" modifies the noun "book" by indicating its location. Modifiers play a crucial role in conveying meaning, clarifying relationships, and adding depth to language. In linguistic analysis, identifying and understanding modifiers can help in parsing sentences, analyzing syntactic structures, and extracting semantic information from text data.[29]

- **Subordinate:** In linguistics, particularly in syntax, "subordinate" refers to a clause or phrase that depends on another clause (typically called the main clause) to form a complete sentence. Subordinate elements add additional information to the main clause and are not complete sentences on their own. They are also known as dependent clauses or phrases[30].
- **Principals:** "Principals" in the context of grammar and syntax refer to the main or central elements of a sentence, including main clauses (independent clauses), subjects, predicates, and key parts of speech.
- **Coordinates:** Coordinated elements in sentences are often linked together by coordinating conjunctions, which include words such as "and", "but", "or", "nor", "for", "yet", and "so". These conjunctions serve to connect elements of equal grammatical importance, allowing for smoother flow and clearer expression of ideas[30].

- **Length of tree:** can be understood as the maximum depth of the parse tree. It measures the longest path from the root of the tree to any leaf (a word that does not govern any other word). This measure gives an indication of the complexity of the sentence's syntactic structure.
- **Max of tree depth**
- **Number of words**
- **Number of characters**
- **Number of tokens**
- **Number of invocabularies:** Using 10000 common words in English.
- **Number of out of vocabularies:** The words that are not in the 10000 common words in English.

These features collectively cover a broad range of linguistic aspects, including lexical richness, syntactic structure, morphological complexity, and readability, which are crucial for determining sentence complexity.

- **Lexical Features:** These features are lexical: (words, characters, tokens, inv-words, oov-words, Average-Syllables) help measure the overall size and lexical complexity of the sentence[31].

- **Syntactic Features:** (Tree-length, tree-max-depth, coordinates, relative-clause-count, subordinate-count, embedded-complement-chains) capture the complexity of the sentence structure[31].

- **Morphological Features:** (Principals, aux-indic-count, modal-indic-count, infinitive-verb-count, noun-count, modifier-count) help in understanding the variety and forms of words used in the sentence[31].

- **Semantic Features:** (Apposition-count, verb-arities-count, num-negative-causal-connectives, num-positive-additive-connectives) help in understanding the meaning and logical structure of the sentence[31].

Using these features, can build a comprehensive model to classify sentences as simple or complex. consider using feature selection techniques to determine which features are the most impactful for the specific classification task.

4) Datasets

4.1 English Dataset

In this project the dataset that is used is from Italian Natural Language Processing Lab. This corpus contains 1,200 English sentences rated by humans with a judgment of complexity. Judgments were collected through a crowd sourcing task in which 20 native speakers of each language were asked to judge how difficult they perceived a given sentence on a complexity scale from 1 (i.e. "very easy") to 7 (i.e. "very difficult"). The datasets of sentences used for the task were taken from two different manually revised treebanks: the automatically converted Wall Street Journal section of the Penn Treebank for the English experiment[32].

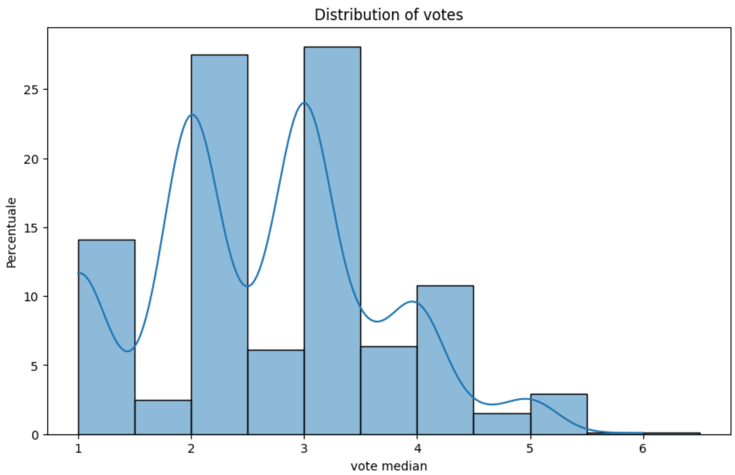

Figure 4: Vote median distribution.

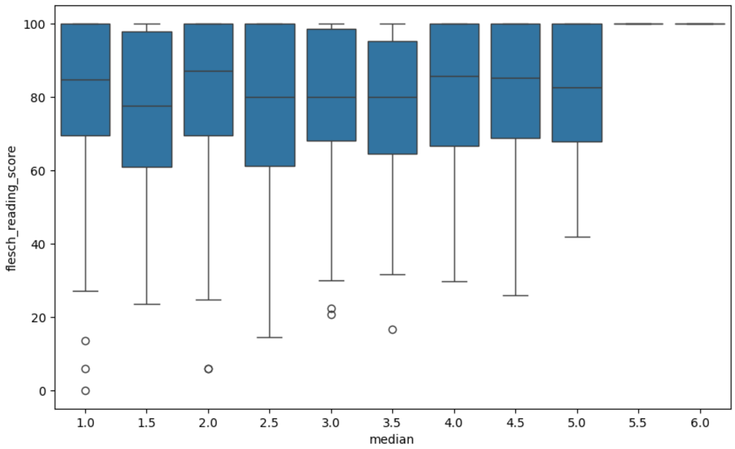

Figure 5: Vote median chart based on score.

Median Scores: The median Flesch reading scores tend to decrease as the median votes increase from 1.0 to 6.0. This suggests that passages associated with higher median votes are generally harder to read.

Distribution Range: The range of Flesch reading scores is quite broad for most vote categories, indicating variability in readability within each vote category.

Outliers: There are a few outliers, particularly for lower vote categories, suggesting some passages are much harder to read compared to the majority.

It was chosen to use the median of the twenty ratings as the reference value for the sentence to decrease the influence of possible outliers. Figure TOP shows the distribution of ratings of the contained sentences and a comparison with Flesch reading scores. It is possible to observe that there is a relationship between Flesch reading scores and the votes cast; lower votes, in fact, correspond to a higher Flesch reading scores. In order to be able to create the two distinct corpora for simple and complex sentences, the dataset was divided by choosing the threshold of 3: sentences with a higher complexity rating were considered complex, below simple. This division results in an imbalance in the dataset, with 939 simple and 260 complex sentences, for a total of 1200 sentences. Sentences missing from the initial dataset were discarded because they were being split into multiple sentences in the spaCy analysis, introducing erroneous data.

4.2 Feature Observation for English dataset

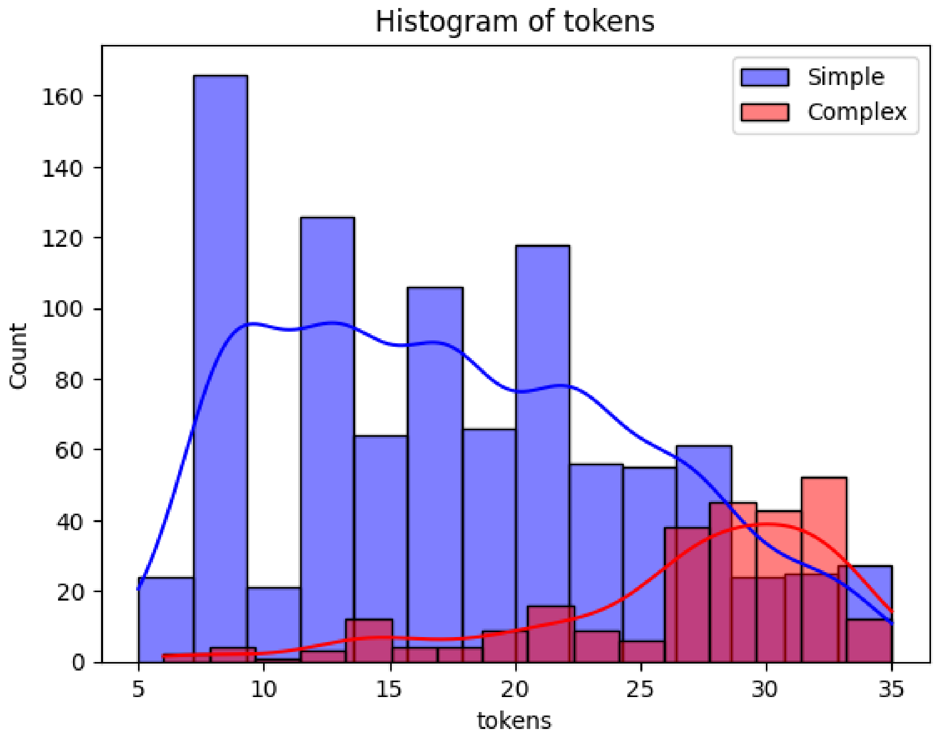

Figure 6: Histogram of tokens

Simple Sentences (Blue): The distribution for simple sentences shows a significant presence of sentences with around 10 tokens, with the count decreasing as the number of tokens increases. This indicates that simple sentences generally have fewer tokens. Complex Sentences (Red): The distribution for complex sentences is more spread out and peaks around 25-30 tokens, suggesting that complex sentences generally have more tokens. The number of tokens is a significant distinguishing feature between simple and complex sentences. Simple sentences tend to have fewer tokens, while complex sentences tend to have more.

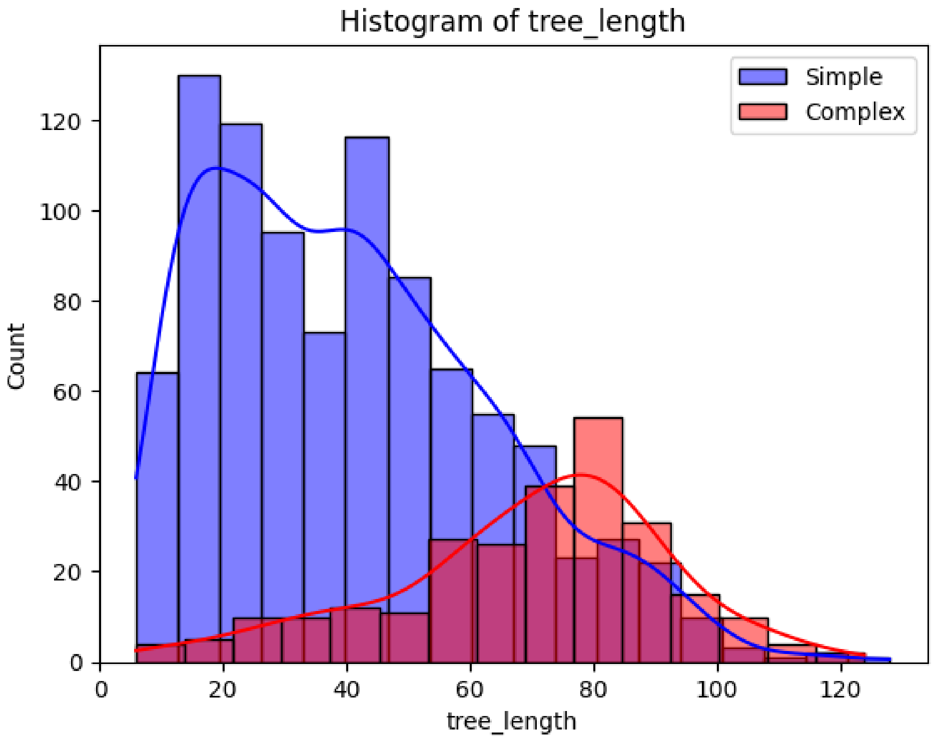

Figure 7: Histogram of length of tree

Simple Trees (Blue): The distribution of simple trees is skewed towards shorter lengths, with most tree lengths falling between 10 and 50 units. The peak is around 20-30 units. Complex Trees (Red): The distribution of complex trees is more spread out with two peaks. One peak is around 20-30 units, similar to simple trees, and another significant peak around 70-80 units. This indicates that complex trees tend to be longer. The length of trees is a distinguishing feature between simple and complex trees. Generally, simple trees tend to be shorter, while complex trees tend to be longer.

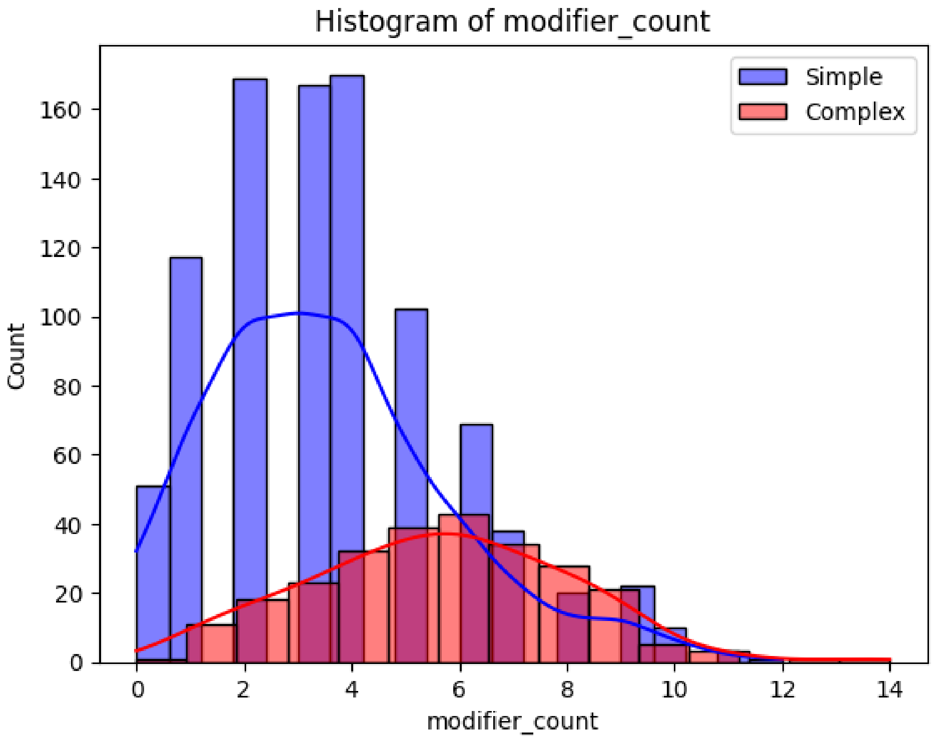

Figure 8: Histogram of modifiers

Simple Sentences (Blue): The distribution of modifier counts in simple sentences is left-skewed, with most simple sentences having between 1 and 4 modifiers. The peak is around 2-3 modifiers. Complex Sentences (Red): The distribution of modifier counts in complex sentences is more spread out and centered around 5-6 modifiers. This indicates that complex sentences tend to have more modifiers than simple sentences. The number of modifiers is a distinguishing feature between simple and complex sentences. Generally, simple sentences tend to have fewer modifiers, while complex sentences tend to have more.

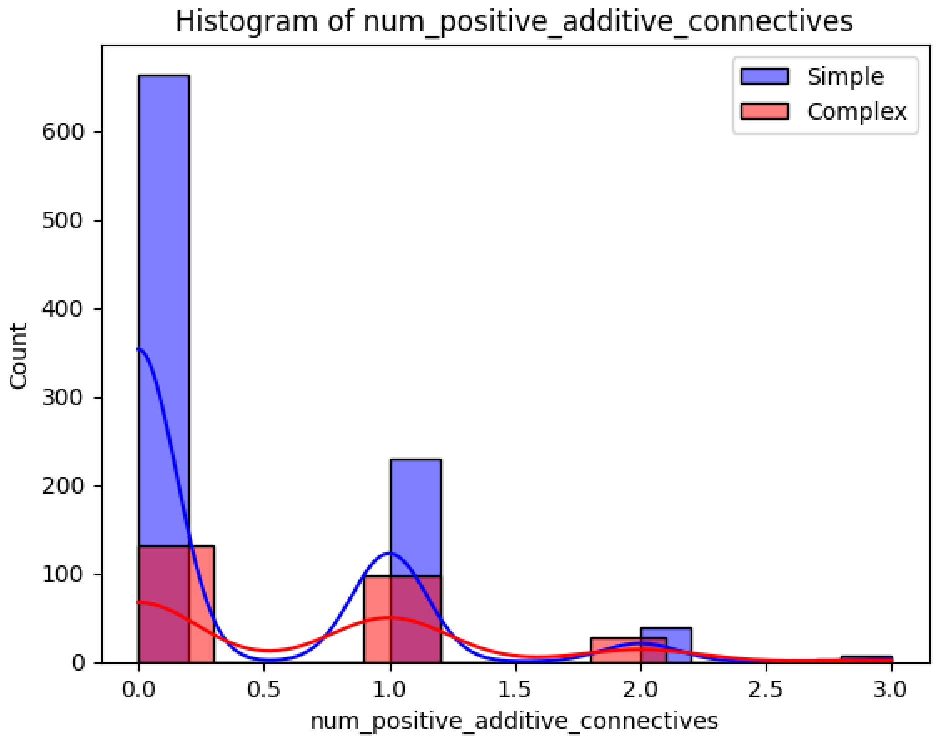

Figure 9: Histogram of additive connective

Simple Sentences (Blue): The distribution of the number of positive additive connectives in simple sentences is highly skewed towards zero, with most simple sentences having no positive additive connectives. There is also a smaller peak at 1. Complex Sentences (Red): The distribution of complex sentences is more spread out but also shows a concentration at zero. However, complex sentences have a higher frequency of having one or more positive additive connectives compared to simple sentences. The number of positive additive connectives is a distinguishing feature between simple and complex sentences. Generally, simple sentences tend to have no or very few positive additive connectives, while complex sentences are more likely to have one or more.

4.3 Italian Dataset

The dataset that is used is from Italian Natural Language Processing Lab. This corpus contains 1123 Italian sentences rated by humans with a judgment of complexity. Judgments were collected through a crowd sourcing task in which 20 native speakers of each language were asked to judge how difficult they perceived a given sentence on a complexity scale from 1 (i.e. "very easy") to 7 (i.e. "very difficult"). The datasets of sentences used for the task were taken from a different manually revised treebanks: the newspaper section of the Italian Universal Dependency Treebank (IUDT) for the Italian experiment[32].

4.4 Feature Observatoion for Italian dataset

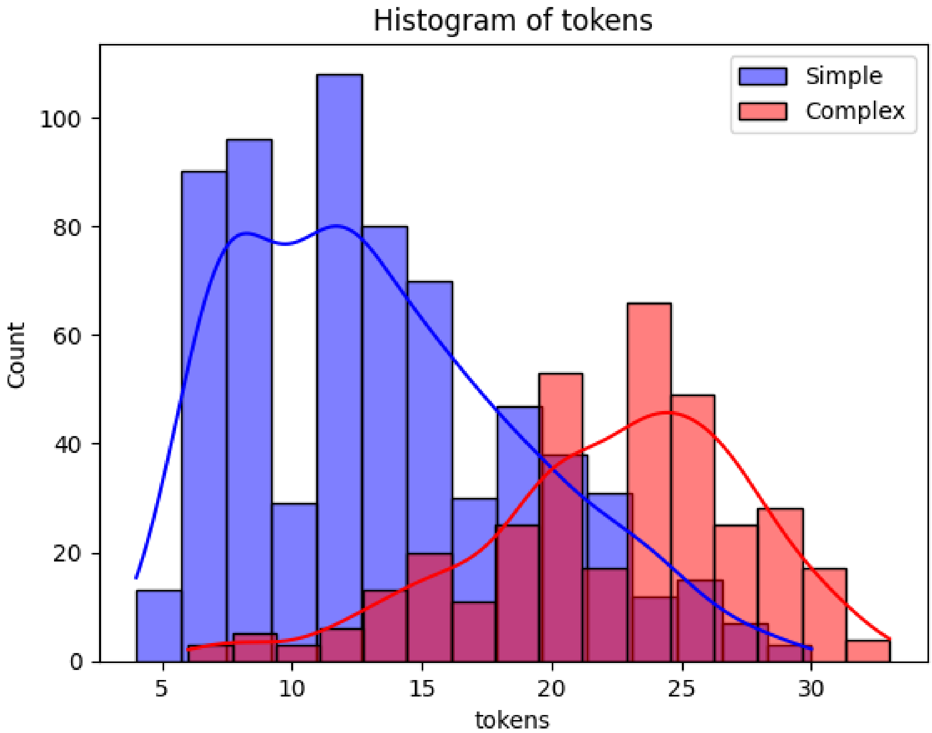

Figure 10: Histogram for tokens

Simple (Blue): The distribution of tokens for the Simple category is skewed towards the left, with most token counts concentrated between 5 and 20. The peak is around 10-15 tokens. Complex (Red): The distribution for the Complex category is more spread out, with a broader peak around 20-25 tokens. This indicates that the token counts for Complex tend to be higher than those for Simple. the result for SVM classifier with the 11 features that are extracted from the first dataset, is shown in the below table. The number of tokens is a distinguishing feature between Simple and Complex categories. Generally, the Simple category tends to have fewer tokens, while the Complex category tends to have more tokens.

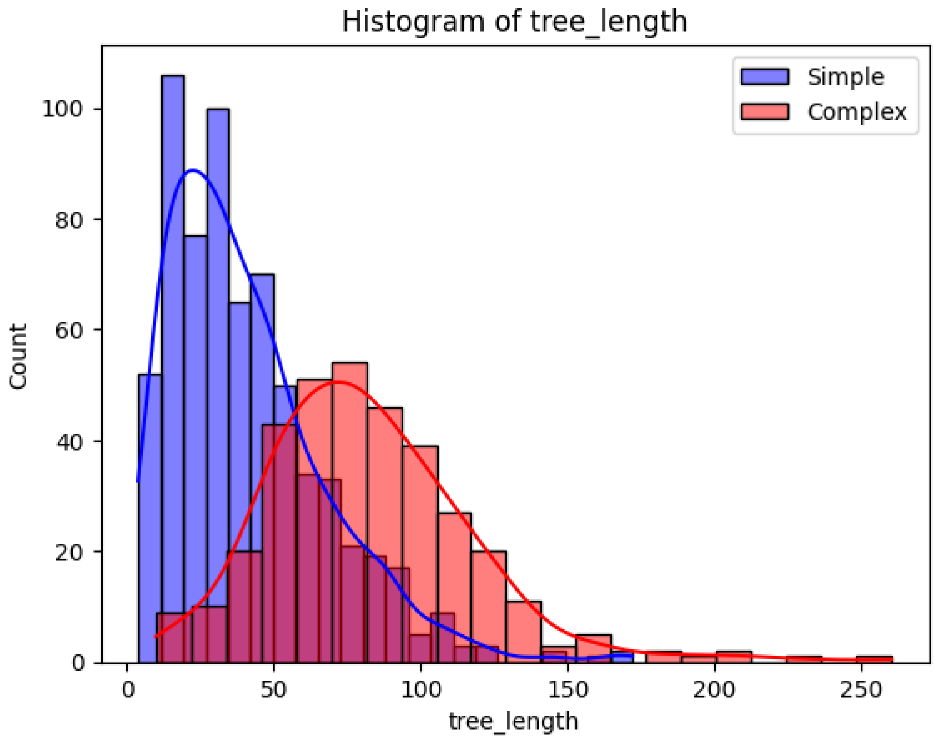

Figure 11: Histogram for tree length

Simple (Blue): The distribution of tree lengths for the Simple category is skewed towards the left, with most tree lengths concentrated between 0 and 50. The peak is around 10-20. Complex (Red): The distribution for the Complex category is more spread out, with a broader peak around 50-70. This indicates that the tree lengths for Complex tend to be longer than those for Simple. The tree length is a distinguishing feature between Simple and Complex categories. Generally, the Simple category tends to have shorter tree lengths, while the Complex category tends to have longer tree lengths.

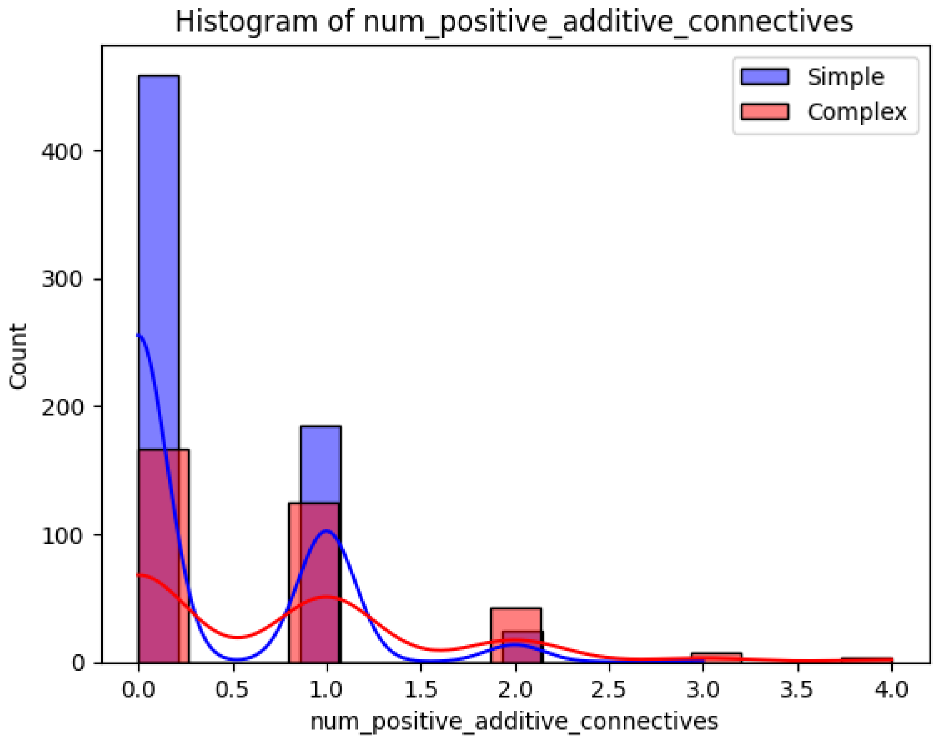

Figure 12: Histogram for additives

Simple (Blue): The distribution of the number of positive additive connectives in the Simple category is heavily skewed towards 0, with the majority of instances having no positive additive connectives. There is a smaller peak at 1 and very few instances with higher counts. Complex (Red): The distribution for the Complex category is more evenly spread out but still has a significant peak at 0. The peak at 1 is more pronounced compared to the Simple category, and there are more instances with higher counts (up to 2 and beyond).The number of positive additive connectives is a distinguishing feature between Simple and Complex categories. Simple sentences predominantly have no positive additive connectives, whereas Complex sentences are more likely to have at least one.

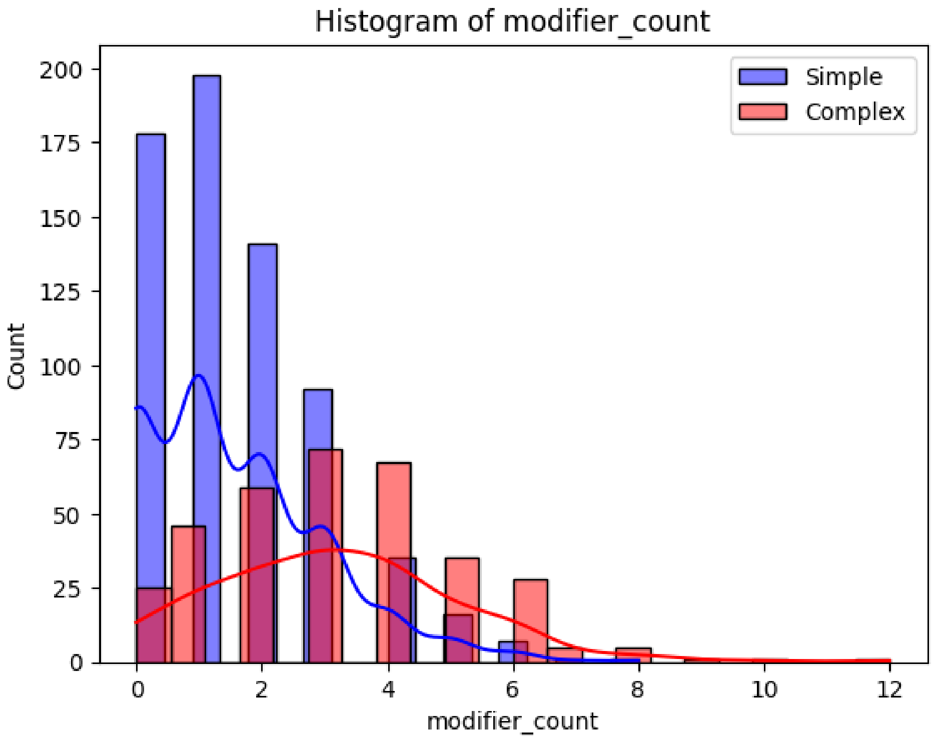

Figure 13: Histogram for modifiers

Simple (Blue): The distribution of the number of positive additive connectives in the Simple category is heavily skewed towards 0, with the majority of instances having no positive additive connectives. There is a smaller peak at 1 and very few instances with higher counts. Complex (Red): The distribution for the Complex category is more evenly spread out but still has a significant peak at 0. The peak at 1 is more pronounced compared to the Simple category, and there are more instances with higher counts (up to 2 and beyond).The number of positive additive connectives is a distinguishing feature between Simple and Complex categories. Simple sentences predominantly have no positive additive connectives, whereas Complex sentences are more likely to have at least one.

4.5 English Dataset using GPT4o

Having only one dataset, as mentioned earlier for the English dataset, is insufficient. Since the results of this research will be used for business purposes, a new dataset is required because the English dataset should be used only for scientific research. Initially, random sentences were extracted from multiple sources, including Wikipedia sentences [33], Wiki sentences [34], Webiss sentences [35], financial dataset sentences [36], and HuggingFace dataset sentences [37]. A total of 2,346 sentences were randomly selected from these datasets. Subsequently, GPT can be utilized to classify this new dataset into simple and complex sentences using prompt engineering.

4.6 Prompt engineering

The datset that is created with random sentences , GPT-4 will be used to classify sentences based on their simplicity or complexity. The key to achieving accurate classifications lies in an effective prompt. The way questions are framed and the structure of the prompt significantly impact GPT-4's ability to differentiate between simple and complex sentences. Therefore, designing the right prompts is essential to ensure that the model generates correct and reliable classifications.
This is the 6 ChatGPT prompt engineering principles from OpenAI that are used for prompting[38]:

- **Clear Instructions:** Providing clear and precise instructions in prompts is crucial for obtaining accurate and relevant responses from AI models. Ambiguity can lead to confusion and less useful outputs. To deal effective prompts, specify the task or question directly and unambiguously. Avoid using vague language and make sure to include all necessary details for the AI to understand the context and requirements.

- **Use Reference Texts:** Incorporating reference texts or examples into prompts can greatly enhance the quality of the AI's response. Reference texts provide context and set a clear standard for the expected output. They help the AI model to better understand the format, tone, and content required, leading to more accurate and relevant responses.

- **Simplify Complex Tasks:** Breaking down complex tasks into smaller, manageable parts can improve the AI's performance. Complex tasks can overwhelm the AI, leading to incomplete or inaccurate responses. By simplifying and segmenting these tasks, you make it easier for the AI to process and generate useful outputs step-by-step.

- **Thoughtful Processing:** Thoughtful processing involves structuring prompts in a way that encourages the AI to think through the task carefully. This can involve asking for step-by-step explanations, detailed reasoning, or incorporating questions that prompt the AI to consider various aspects of a topic. This approach can enhance the depth and quality of the AI's responses.

- **Use External Tools:** Leveraging external tools and resources can augment the capabilities of the AI. This might involve using APIs, databases, or other software to provide additional data, perform specific functions, or validate information. Integrating these tools can extend the AI's functionality and improve the accuracy and relevance of its outputs.

- **Systematic Testing:** Systematic testing involves continuously evaluating and refining your prompts to optimize their performance. This process includes experimenting with different phrasings, formats, and structures, and analyzing the results to identify the most effective approaches. By systematically testing and iterating on your prompts, you can significantly enhance the quality and consistency of the AI's responses.

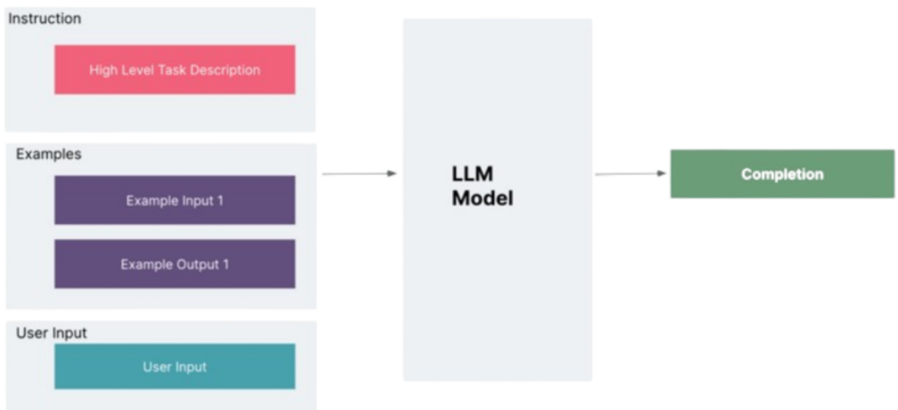

Figure 14: composition of a typical prompt

4.7 Classify the Dataset with GPT4o

Before using GPT4o to classify simple and complex sentences, it is necessary to explain to GPT4o why certain sentences are considered simple and others complex.

4.8 Prompt for Identifying Criteria in Simple and Complex Sentences

At first 30 sentences from simple sentences and 30 sentences from complex sentences are used to train the GPT4o. these sentences are extracted randomly form the first dataset that is used for extracting features. For each sentence created reasons why the sentence is simple and why the sentence is complex.

4.9 simple sentence samples for GPT4o

Below there are Some sample of simple sentences that are used for training GPT4o and why these sentences are simple:

- **Sentence:** "The agency will withdraw from the production of nine food products, maintaining production of the two most important ones, corn and milk."
  Reason: This sentence is straightforward for native speakers because it follows a common structure, uses familiar vocabulary, and isn't overly long.

- **Sentence:** "The company, which is retaining most of its wine and all of its soft-drink interests, didn't break out results for the businesses it plans to sell."
  Reason: It presents information in a clear and direct manner. It describes what the company is doing, is not overly long, and has a straightforward structure.

- **Sentence:** "In 1982, he started a factory in Greece."
  Reason: This sentence is straightforward for native speakers because it presents a clear and concise piece of information.

- **Sentence:** "In 1986, Mr. Simmons also served on a committee of businessmen headed by William Seidman, chairman of the Federal Deposit Insurance Corp. and the Resolution Trust Corp."
  Reason: It provides clear chronological information and straightforward details about Mr. Simmons' involvement. The sentence structure is uncomplicated and not overly long.

- **Sentence:** "He cites IBM, which reported a 30% earnings decline in the third quarter, and which last week announced a $1 billion buy-back of its shares."
  Reason: It's clear, not overly long, uses common vocabulary, and has a straightforward structure without complex syntactic elements.

4.10 complex sentence samples for GPT4o

Below there are Some sample of complex sentences that are used for training GPT4o and why these sentences are complex:

- **Sentence:** "In the nine months, net rose 35% to $120.1 million, or $1.64 a share, from $89.2 million, or $1.22 a share, a year earlier."
  Reason: The sentence includes numerical data, financial terminology, a comparative structure, and slightly convoluted syntax.

- **Sentence:** "The housing starts numbers, however, are one of the least precise of the government's economic indicators and are often revised significantly as more information is collected."
  Reason: The sentence discusses a specialized topic (government economic indicators), uses comparative language ("one of the least precise"), employs complex syntax, and introduces technical terminology.

- **Sentence:** "Yesterday, the spokeswoman said sales of Delmed products through the exclusive arrangement with National Medical accounted for 87% of Delmed's 1988 sales of $21.1 million."
  Reason: The sentence contains a combination of temporal reference, numerical data, complex syntax, and technical language.

- **Sentence:** "However, Delta Air Lines fell 1 1/2 to 67 1/2 and USAir Group dropped 3/4 to 42 1/2."
  Reason: The sentence includes abbreviated numerical notation, multiple company references, technical language, and specific numerical data.

- **Sentence:** "Although a substantial short position reflects heavy speculation that a stock's price will decline, some investors consider an increase in short interest bullish because the borrowed shares eventually must be bought back."
  Reason: The sentence uses a combination of complex sentence structure, technical terminology, conceptual complexity, and contrasting ideas.

Below is the Prompt that is used for finiding the criteria:

Prompt: The Fascinating World of Simple & Complex Sentences.

Let's dive into the intricacy of the English language. Your assignment is to categorize sentences into two types - 'Simple' and 'Complex.' To begin with, you will be given a training set that encompasses sentences from both categories, along with explanations detailing why they fall into their respective classification. Your responsibility is to thoroughly scrutinize these examples to unearth the differentiating factors between simple and complex sentences.

Afterward, you will be shared with a variety of sentences that are yet to be classified. Leveraging the criteria extracted from the training set, you will now have to assign these sentences into their fitting category - simple or complex. But remember, your classification decisions must align with the guidelines you've derived from the training data, and for classify new sentences you should prioritize the criteria of simple sentences rather than complex sentences for classify new sentences

For every decision you make, provide a rationale explaining how each sentence meets the criteria of being simple or complex. Your explanations should be comprehensive, crisp, and must connect directly with the parameters you extracted initially. Great ready to decode English sentences, and remember to communicate your findings in English.

After following paragraph, GPT4o was given the sentences as simple and complex, along with the reasons mentioned above. Based on this input, GPT4o provided the following criteria for distinguishing between simple and complex sentences.

4.11 Criteria for Simple Sentences based on GPT4o:

After training GPT4o with simple and complex sentences, it provided the following criteria for identifying simple sentences. The sentences listed under each criterion serve as examples, illustrating why they are considered simple.

- **Clear and Direct Information:**
  Presents information in a straightforward manner without requiring additional context.
  Examples:
  - "In 1982, he started a factory in Greece."
  - "Armstrong's shares, also listed on the Big Board, closed at $39.125, up 12.5 cents."
  - "Of course, that isn't really the case."
  - "He cites IBM, which reported a 30% earnings decline in the third quarter, and which last week announced a $1 billion buy-back of its shares."
  - "Revco received the offer Oct. 20, but issued a response yesterday only after a copy of the proposal was made public by bondholders."
- **Common Vocabulary:**
  Uses familiar and easily understood words and phrases.
  Examples:

  - "Betty Raptopoulos, senior metals analyst at Prudential-Bache Securities in New York, agreed that most of the selling was of a technical nature."
  - "Mr. Icahn advocates the sale of the company's steel operations, and Mr. Corry doesn't necessarily disagree."
  - "The announcement, made after the close of trading, caught analysts by surprise."
  - "Field offices at New Orleans; Houston; Denver; Midland, Tex.; Bakersfield, Calif.; Oklahoma City; and Liberal, Kan., will be maintained."
  - "But investment bankers say that stock market uncertainties in the U.S. may cause many European companies to mark time before bidding for American companies, in the hope that share prices will come down."

- **Straightforward Structure:**
  Follows a basic structure (subject-verb-object) without complex subordination.
  Examples:
  - "The announcement, made after the close of trading, caught analysts by surprise."
  - "Betty Raptopoulos, senior metals analyst at Prudential-Bache Securities in New York, agreed that most of the selling was of a technical nature."
  - "As with many other goods, the American share of Japan's PC market is far below that in the rest of the world."
  - "Exxon's profitability, like that of many other oil companies, was hurt during the third quarter by declining returns from the chemicals and refining and marketing businesses."
  - "At one point in the documentary, Mr. Sohmer, who is Jewish, says he felt rejected by many of the Protestants and Southerners he met at Yale."

- **Not Overly Long:**
  Tends to be concise and not overly lengthy.
  Examples:
  - "Armstrong's shares, also listed on the Big Board, closed at $39.125, up 12.5 cents."

  - "Field offices at New Orleans; Houston; Denver; Midland, Tex.; Bakersfield, Calif.; Oklahoma City; and Liberal, Kan., will be maintained."
  - "But the Japanese Fisheries Association criticized moves to ban the practice in international waters."
  - "Elsewhere, share prices closed lower in Zurich, Amsterdam, Milan and Stockholm."

- **Logical Flow:**
  Presents information in a logical and sequential manner.
  Examples:

  - "Field offices at New Orleans; Houston; Denver; Midland, Tex.; Bakersfield, Calif.; Oklahoma City; and Liberal, Kan., will be maintained."
  - "As a result of the refinancing, the company said the interest on the debt will fall to slightly more than 11% from slightly more than 14%."
  - "Now he has moved to Oklahoma where costs are lower, and started a new company, Adsi Inc., to market his machine."
  - "These and other problems squeezed Federal's profit margins last year to 8%, down from more than 13% annually in the first half of the decade."

- **No Complex Syntactic Elements:**
  Does not include multiple clauses or intricate grammatical constructions.
  Examples:

  - "As a result of the refinancing, the company said the interest on the debt will fall to slightly more than 11% from slightly more than 14%."
  - "Betty Raptopoulos, senior metals analyst at Prudential-Bache Securities in New York, agreed that most of the selling was of a technical nature."
  - "In London, the Financial Times 100-share index finished 30.1 points higher at 2112.2."
  - "The U.S. ranks fourth of countries that have concluded joint ventures, behind West Germany, Finland and Italy."

- "Following the release of the company's fourth-quarter earnings, Apple Computer dropped 3/4 to 48 on volume of more than 2.3 million shares."

4.12 Criteria for Complex Sentences based on GPT4o:

After training GPT4o with simple and complex sentences, it provided the following criteria for identifying complex sentences. The sentences listed under each criterion serve as examples, illustrating why they are considered complex.

- **Complex Sentence Structure:**
  Often includes multiple clauses, subordination, or compound structures.
  Examples:
  - "Although a substantial short position reflects heavy speculation that a stock's price will decline, some investors consider an increase in short interest bullish because the borrowed shares eventually must be bought back."
  - "If the Legislature doesn't repeal the law, due for revision in 1990, Mr. Jones says Humana may move its insurance operations, including 3,000 jobs in Louisville, to another state."
  - "Instead, many small and medium-sized banks, and some larger ones, are likely to take one of the other two options open to them under the plan, Japanese banking officials said."
  - "Hani Zayadi was appointed president and chief executive officer of this financially troubled department store chain, effective Nov. 15, succeeding Frank Robertson, who is retiring early."
- **Technical Terminology:**
  Contains specialized or technical language that might require specific knowledge to understand.
  Examples:
  - "The housing starts numbers, however, are one of the least precise of the government's economic indicators and are often revised significantly as more information is collected."
  - "Here are the seasonally adjusted changes in the components of the Labor Department's consumer price index for September."
  - "Instead, the GAO and the Congressional Budget Office said, the RTC should consider using Treasury debt, which is less expensive and subject to oversight by Congress."

  - "Some House Democrats are trying to head off an appointment by President Bush to the board that oversees the savings-and-loan bailout, contending that the prospective nominee is the head of troubled banks himself."

- **Inclusion of Numerical Data:**
  Frequently incorporates specific figures, statistics, or financial data.
  Examples:

  - "In the nine months, net rose 35% to $120.1 million, or $1.64 a share, from $89.2 million, or $1.22 a share, a year earlier."
  - "Yesterday, the spokeswoman said sales of Delmed products through the exclusive arrangement with National Medical accounted for 87% of Delmed's 1988 sales of $21.1 million."
  - "Dow Jones industrials 2603.48, up 6.76; transportation 1191.86, up 1.43; utilities 216.74, up 0.88."
  - "Fidelity's junk fund has fallen 2.08% this year through Oct. 19, Lipper says; the Vanguard fund rose 1.84%; and the T. Rowe Price fund edged up 0.66%."

- **Detailed Information:**
  Provides dense information, often combining several pieces of data or aspects within a single sentence.
  Examples:

  - "Hani Zayadi was appointed president and chief executive officer of this financially troubled department store chain, effective Nov. 15, succeeding Frank Robertson, who is retiring early."
  - "Lynch Corp. said its Lynch Telephone Corp. subsidiary completed the acquisition of Western New Mexico Telephone Co. for $20 million plus assumption of $24 million of debt."
  - "Digital, based in Maynard, Mass., hopes to stage a repeat performance in mainframes, and it has spent almost $1 billion developing the new technology."
  - "The suits relate to a $200 million loss, disclosed in December, that was suffered by West Virginia's consolidated investment pool."

- **Specialized Context:**
  Refers to specific contexts, such as legal, financial, or technical fields.
  Examples:

- "Here are the seasonally adjusted changes in the components of the Labor Department's consumer price index for September."
- "The housing starts numbers, however, are one of the least precise of the government's economic indicators and are often revised significantly as more information is collected."
- "The GAO and the Congressional Budget Office said, the RTC should consider using Treasury debt, which is less expensive and subject to oversight by Congress."
- "Mr. Holmes was the subject of a page one profile in The Wall Street Journal in 1984, after the SEC questioned him about ties between him and companies he touted in a newsletter."

- **Comparative and Contrasting Ideas:**
  Presents comparisons, contrasts, or cause-effect relationships within the same sentence.
  Examples:
  - "Although a substantial short position reflects heavy speculation that a stock's price will decline, some investors consider an increase in short interest bullish because the borrowed shares eventually must be bought back."
  - "To justify their exempt status and avoid penalties, these businesses must show once a year that each and every transaction on which they didn't pay sales tax was a legitimate business expense."
  - "If the Legislature doesn't repeal the law, due for revision in 1990, Mr. Jones says Humana may move its insurance operations, including 3,000 jobs in Louisville, to another state."
  - "Instead, many small and medium-sized banks, and some larger ones, are likely to take one of the other two options open to them under the plan, Japanese banking officials said."

4.13 Prompt for classifying Simple and Complex Sentences

Below is the Prompt that is used for classifying Simple and Complex Sentences:

Prompt: I need your assistance in classifying sentences based on their structure. Below are the criteria for simple and complex sentences, with multiple examples for each criterion. Please read the criteria carefully and classify the given sentences accordingly and slightly prioritize the simple sentence criteria for classification of the sentences. You must adhere to provided criteria.

Criteria for Simple Sentences:
Provide the criteria mentioned above for simple and complex sentences, and instruct GPT to perform the task accordingly.

4.14 Evaluating the Prompt for Accurate Classification of Simple and Complex Sentences

Using these criteria, the sentences from the initial dataset were provided to GPT4o to evaluate its performance. Upon applying the criteria to the existing sentences, GPT4o correctly classified 90% of the complex sentences and achieved 90% accuracy for the simple sentences.
Even when using the Flesch Reading Score for readability assessment, the average score for complex sentences is 46, while for simple sentences it is 59.87. This indicates that GPT4o is functioning correctly in distinguishing between simple and complex sentences.
90-100: Very easy to read. Easily understood by an average 11-year-old student.
80-89: Easy to read. Conversational English for consumers.
70-79: Fairly easy to read.
60-69: Plain English. Easily understood by 13- to 15-year-old students.
50-59: Fairly difficult to read.
30-49: Difficult to read. Best understood by college graduates.
0-29: Very difficult to read. Best understood by university graduates.
Now for checking other metrics that they are checked whether the GPT4o is correct or not, so the most important features are calculated for the median and it is reached to these numbers as below:

The comparison of features between simple and complex sentences shows that complex sentences generally have higher values across several metrics. For instance, the number of words and characters in complex sentences is approximately double that of simple sentences, indicating longer and potentially more detailed sentences. The tokens count, which includes words and punctuation, is also higher in complex sentences, reflecting their increased complexity. Additionally, complex sentences have more modifiers and out-of-vocabulary (OOV) words, suggesting more detailed and varied language. Interestingly, while both sentence types have no subordinates or embedded complement chains, complex sentences feature appositions and positive additive connectives that are absent in simple sentences. The tree length, which represents the syntactic structure's depth, is significantly greater in complex sentences, highlighting their more intricate grammatical construction. Overall, the data clearly illustrate that complex sentences are richer and more elaborate than simple sentences.

Feature	Simple	Complex
Number of words	11.0	22.0
Number of characters	53.0	104.0
Tokens count	12.0	23.0
Number of subordinates	0.0	0.0
Tree length	21.0	52.0
Number of modifiers	2.0	5.0
Number of OOV words	0.0	1.0
Number of INV words	10.0	17.0
Number of apposition counts	0.0	1.0
Number of embedded complement chains count	0.0	0.0
Number of positive additive connectives	0.0	1.0

Table 1: Comparison of Features between Simple and Complex Sentences

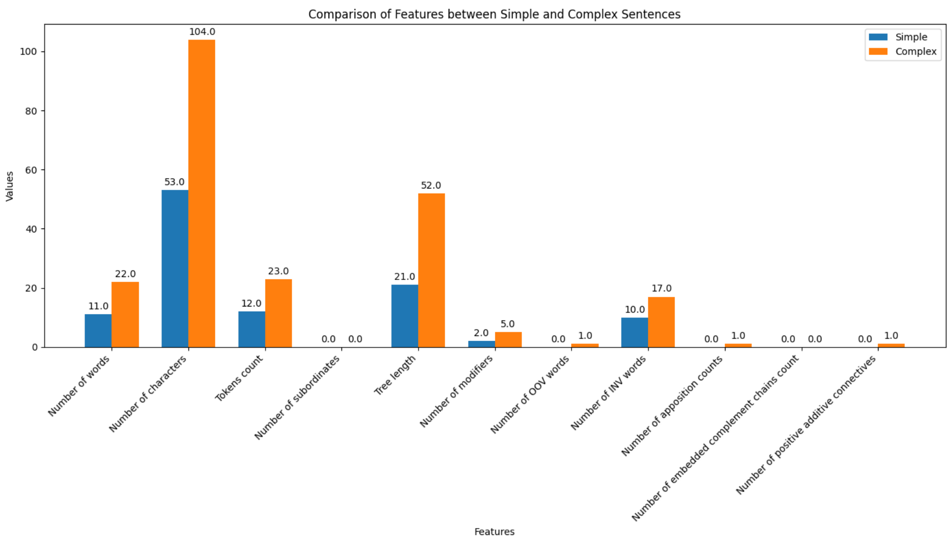

Figure 15: Comparison of Features between Simple and Complex Sentences

4.15 Feature Observatoion for GPT4o dataset

showing some histograms for important features.

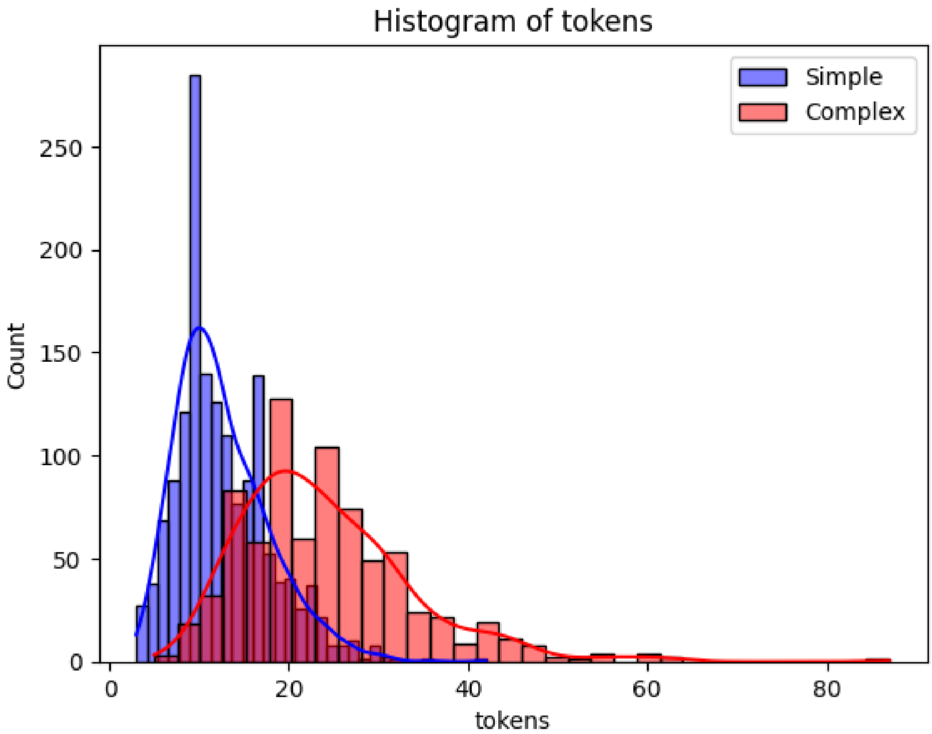

Figure 16: Histogram for tokens

Simple Sentences (Blue): The distribution of simple sentences is skewed towards the left, with most sentences having fewer tokens. The peak is around 10 words. Complex Sentences (Red): The distribution of complex sentences is more spread out and has a broader peak around 15-20 tokens. This indicates that complex sentences tend to be longer. Length as a Feature: The number of tokens is a distinguishing feature between simple and complex sentences. Generally, simple sentences tend to be shorter, while complex sentences tend to be longer.

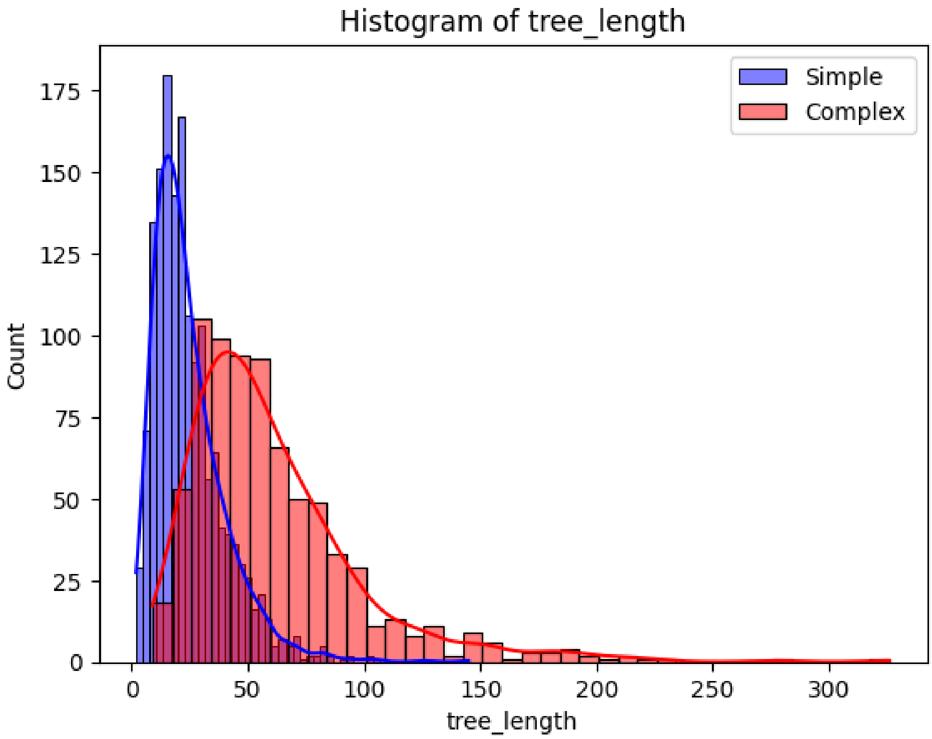

Figure 17: Histogram for tree length

Simple Sentences (Blue): The distribution of tree_length for simple sentences is skewed towards the left, with most simple sentences having shorter tree lengths. The peak is around 20-30 nodes. Complex Sentences (Red): The distribution for complex sentences is more spread out and skewed towards the right, indicating longer tree lengths. The peak is around 40-50 nodes, but the distribution extends further, with some sentences having tree lengths over 200 nodes. Tree Length as a Feature: The length of the syntactic tree is a strong distinguishing feature between simple and complex sentences. Simple sentences generally have shorter syntactic trees, while complex sentences tend to have longer ones.

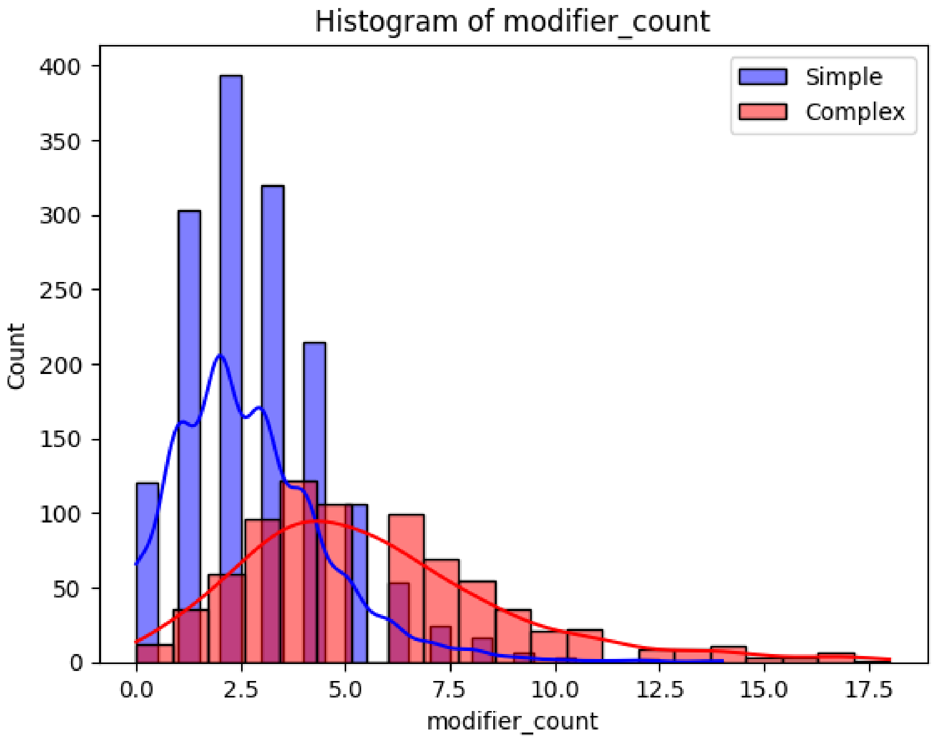

Figure 18: Histogram for modifiers

Simple Sentences (Blue): The distribution for simple sentences peaks around 2-3 modifiers and drops off significantly beyond 5 modifiers. This indicates that simple sentences generally have fewer modifiers. Complex Sentences (Red): The distribution for complex sentences is broader, with a peak around 4-5 modifiers and a longer tail, indicating that complex sentences often have more modifiers. The count extends beyond 10 modifiers.

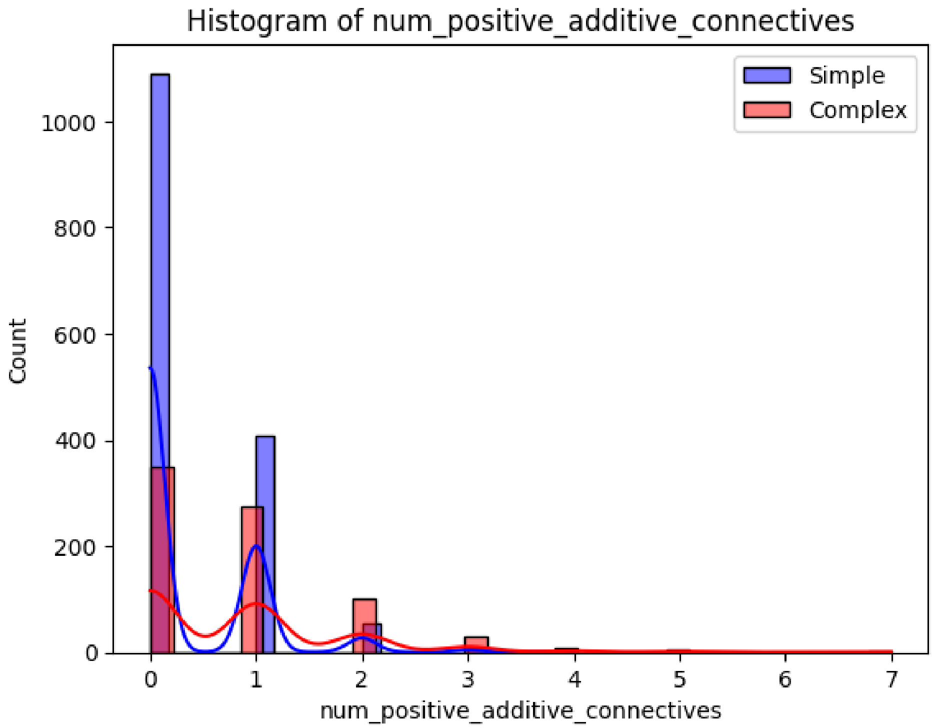

Figure 19: Histogram for additives

Simple Sentences (Blue): The vast majority of simple sentences have 0 positive additive connectives, as indicated by the significant peak at 0. There are smaller peaks at 1 and 2 connectives, but very few simple sentences have more than 2. Complex Sentences (Red): Complex sentences also have 0 positive additive connectives, but with a smaller peak compared to simple sentences. There are noticeable counts at 1 and 2 connectives, and a few complex sentences have up to 3 or more. Combine Features: Use the number of positive additive connectives in combination with other features (such as sentence length, tree length, and modifier count) can improve the classification model.

5) Feature selection and Implementation

5.1 English Dataset

Recursive Feature Elimination (RFE) is a feature selection technique that helps to select the most relevant features for building a machine learning model. When combined with a Support Vector Machine (SVM) classifier, RFE can enhance the model's performance by eliminating irrelevant or redundant features[27]
the Steps are:

- Fit the Model: Train the model on the initial set of features.
- Rank Features: Evaluate the importance of each feature using a chosen metric.
- Eliminate Least Important Features: Remove the least important feature(s).
- Repeat: Refit the model and repeat the process until the desired number of features remains.

After the RFE with the SVM classifier, it reached to 13 features that are most important than others, and accuracy reached to 87.92 percent accuracy and f1-score of 0.7888.

5.2 Biserial Correlation Coefficient

The biserial correlation coefficient is used to measure the relationship between a continuous variable and a binary variable. In the context of binary classification, such as distinguishing between simple and complex sentences, the biserial correlation can help to understand how a continuous feature (e.g., sentence length) correlates with the binary classification outcome (simple vs. complex)[39].

Features
Words
Characters
Tokens
Inv_words
Oov_words
Tree_length
Average_syllables
Apposition_count
Verb_arities_count
Num_positive_additive_connectives
Embedded_complement_chains
Modifier_count
Subordinate_count

Table 2: selected Features

$$r_{pb} = \frac{\bar{X}_1 - \bar{X}_0}{s} \times \frac{p_1 \cdot p_0}{\phi(z)}$$

- $\bar{X}_1$: Mean of the continuous variable for the binary class labeled as 1.
- $\bar{X}_0$: Mean of the continuous variable for the binary class labeled as 0.
- s: Standard deviation of the continuous variable.
- p_1: Proportion of cases in class 1.
- p_0: Proportion of cases in class 0.
- $\phi(z)$: The standard normal probability density function evaluated at z, which is the point of division for the binary variable.

The biserial correlation is suitable for this case for several reasons:

- Binary Classification: there is a binary classification problem (simple vs. complex sentences).
- Continuous Features: there are continuous features (e.g., sentence length) that can be correlated with the binary outcome.
- Interpretability: Biserial correlation can provide insights into how strongly a continuous feature is associated with the binary classification outcome, which can help in feature selection and model interpretation.

below is the table for the Biserial correlation coefficient.

Feature	**Value**
Words	0.396
Characters	0.402
Tokens	0.474
Inv_words	0.351
Oov_words	0.175
Tree_length	0.457
Average_syllables	-0.006
Apposition_count	0.169
Verb_arities_count	-0.030
Num_positive_additive_connectives	0.178
Embedded_complement_chains	0.0798
Modifier_count	0.349
Subordinate_count	0.146

Table 3: Biserial correlation for the features

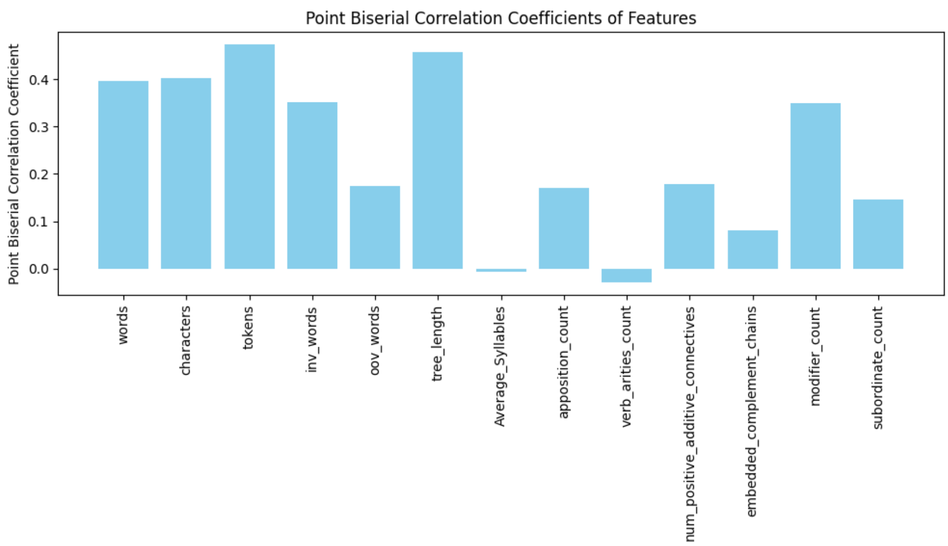

Figure 20: Biserial correlation for the features for English dataset

After using the biserial correlation for this 13 features, two features have low absolute value of correlation: 1) average_syllables 2) verb_arities_count. After removing these features, the accuracy and F1-score for this dataset remain unchanged. As discussed below, even in the updated dataset, these two features not only fail to improve accuracy but also contribute to a decrease in both accuracy and the F1-score.

- Redundancy and Noise Reduction: Features with nearly zero correlation to the target variable typically do not provide useful information for the classifier. Removing them can reduce noise and redundancy in the dataset, potentially improving the classifier's performance on unseen data.
- Model Simplicity: Simplifying the model by reducing the number of features can make it more interpretable and reduce the risk of overfitting. This is particularly important when applying the model to new datasets, as it generalizes better with fewer but more relevant features.
- Performance Validation: The fact that the model with 11 features has shown better accuracy and F1 score on another dataset suggests that the two features you removed were not contributing to the model's performance. This indicates that the remaining 11 features are sufficient and potentially more robust for generalization.
- Statistical Evidence: Biserial correlation is a useful method to determine the relevance of features. Features with near-zero correlation are statistically unlikely to be informative for the classification task.

Removing features based on biserial correlation coefficient after performing Recursive Feature Elimination (RFE) is absolutely normal. There are key points why these removing are reasonable:

- Cross validation : Using GridSearchCV for validation to find best parameters and with 5 folds and the parameters are: C =100, kernel = Linear. When performing cross-validation with a Support Vector Machine (SVM) using the parameters C = 100, and a linear kernel, the model is configured to prioritize minimizing classification errors on the training set. The high value of C (100) indicates a strict penalty for misclassification, which can lead to a model that may potentially overfit the training data, especially if the data contains noise or outliers. The linear kernel assumes that the data is linearly separable, meaning that the relationship between the features and the target variable can be captured by a straight line (or hyperplane in higher dimensions).
- Complementary Methods: RFE and biserial correlation are complementary feature selection methods. RFE evaluates the importance of features based on their contribution to the classifier's performance, while biserial correlation assesses the direct relationship between each feature and the target variable.

- Empirical Validation: The observation that the model performs equally well (or better) on another dataset after removing the two features with nearly zero correlation provides empirical validation. This suggests that those features were not contributing meaningfully to the classifier's performance and that their removal helps in improving the model's generalization capability.

- Improved Generalization: By removing features with negligible correlation to the target, reduce the risk of overfitting and enhance the model's ability to generalize to new data. This is supported by the improved accuracy and F1 score on another dataset.

- Feature Relevance: Combining RFE with biserial correlation ensures that retain features that are both important for the classifier's performance and have a meaningful relationship with the target variable. This two-step feature selection process enhances the robustness of the feature set.

After performing Recursive Feature Elimination (RFE) and checking the results with Biserial correlation, it was found that average_syllables and verb_arities_count have no correlation with the classifier. Therefore, these features were removed. The accuracy and F1-score remained unchanged whether these features were included or not, indicating that removing them can help reduce redundancy and minimize the risk of overfitting without affecting the model's performance.

	precision	**recall**	**f1-score**	**support**
0	0.91	0.95	0.93	939
1	0.76	0.69	0.78	260
accuracy	0.88			
macro avg	0.82	0.77	0.79	1119
weighted avg	0.87	0.88	0.87	1119

Table 4: Classification Report

The other models are used for this classification to see which one is better. other models are Decision Tree, Logistic regression, Random Forest.

5.3 Decision Tree classifier

For the decision tree classifier, RFE was used for feature selection, with 5-fold cross-validation and accuracy as the scoring metric, using entropy as the criterion. Interestingly, the features selected by RFE were identical to those extracted by the SVM model.potential reasons for the same features can be due to the most important features for classification may be highly correlated with the target variable, which would make them equally significant for both models. Another reason could be there are features in the dataset that are either redundant (highly correlated with other features) or irrelevant (provide little to no information for classification), RFE will likely remove these features for both models. This can result in the same subset of key features being selected for both, so, the similarity in feature selection may indicate that the features identified by RFE are the most informative for your specific dataset, regardless of the underlying model.

5.4 Logistic regression classifier

For logistic regression, RFE was applied again for feature selection, resulting in only one feature change: Embedded_complement_chains was removed, and number_of_subordinate was added to the features. The hyperparameters used for the classifier are as follows:

penalty: l2 (Using L2 regularization to prevent overfitting) C: 1.0 (Regularization strength; 1.0 is the default) solver: lbfgs (A robust solver suitable for this classification task) max_iter: 200 (Increased from the default to ensure convergence) tol: 1e-4 (Tolerance for stopping criteria)

5.5 Random Forest classifier

For the Random Forest classifier, feature importance was used to identify the most relevant features. After carefully considering the features with higher importance scores, the only difference compared to the SVM model is the inclusion of number_of_subordinate. All other features remain the same. The hyperparameters used for the Random Forest classifier are as follows:

n_estimators: Number of trees in the forest (set to 1 in the code). max_features: Number of features to consider when splitting a node ('auto' uses all, sqrt uses the square root of the features). max_depth: Maximum depth of the trees, controlling tree growth (values from 10 to 110, plus None for no limit). min_samples_split: Minimum number of samples required to split a node (values 2, 5, 10). min_samples_leaf: Minimum number of samples required at a leaf node (values 1, 2, 4). bootstrap: Whether bootstrap sampling is used when building trees (True or False).

5.6 Result

Based on the performance metrics provided for each classifier, SVM (Support Vector Machine) performing better compared to Decision Tree, Logistic Regression, and Random Forest. SVM has the highest accuracy among all models, indicating that it correctly classifies sentences more often than the other classifiers. The F1-score, which balances precision and recall, is also highest for SVM, indicating better overall performance in terms of both false positives and false negatives. And Also SVM are effective in high-dimensional spaces and can handle complex relationships between features. This is beneficial for classifying sentences, which can vary widely in structure and content. SVM aims to maximize the margin between different classes, which generally leads to better generalization and lower overfitting, especially when compared to decision trees or random forests which can easily overfit complex data. SVM are less sensitive to noise in the data compared to decision trees and random forests. They achieve this by maximizing the margin, focusing on the support vectors (instances closest to the decision boundary), which helps in ignoring outliers or noisy data points. SVM can perform well even with relatively small datasets,. This characteristic is crucial if dataset is not very large.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
SVM	0.88	0.82	0.77	0.78
Decision Tree	0.80	0.72	0.71	0.51
Logistic Regression	0.87	0.81	0.79	0.66
Random Forest	0.82	0.69	0.71	0.53

Table 5: Comparison of Classification Models

5.7 GPT4o Dataset

Using the previous dataset, 11 key features were identified. As mentioned earlier, since the results of this analysis will be applied for business purposes, it is essential to test these features on a different dataset to determine their generalizability. An SVM classifier was applied to the new dataset, utilizing GridSearchCV with 5-fold cross-validation to optimize the model.

Additionally, the two features previously removed based on the Biserial correlation coefficient were also excluded from the new feature set, leaving 11 features for testing on the new dataset.

The table below presents the Biserial correlation coefficient results for the new dataset, providing insight into the relationship between the features and the classification task.

Feature	Value
Words	0.61
Characters	0.58
Tokens	0.61
Inv_words	0.58
Oov_words	0.24
Tree_length	0.56
Apposition_count	0.003
Num_positive_additive_connectives	0.28
Embedded_complement_chains	0.11
Modifier_count	0.50
Subordinate_count	0.25

Table 6: Biserial correlation for the features

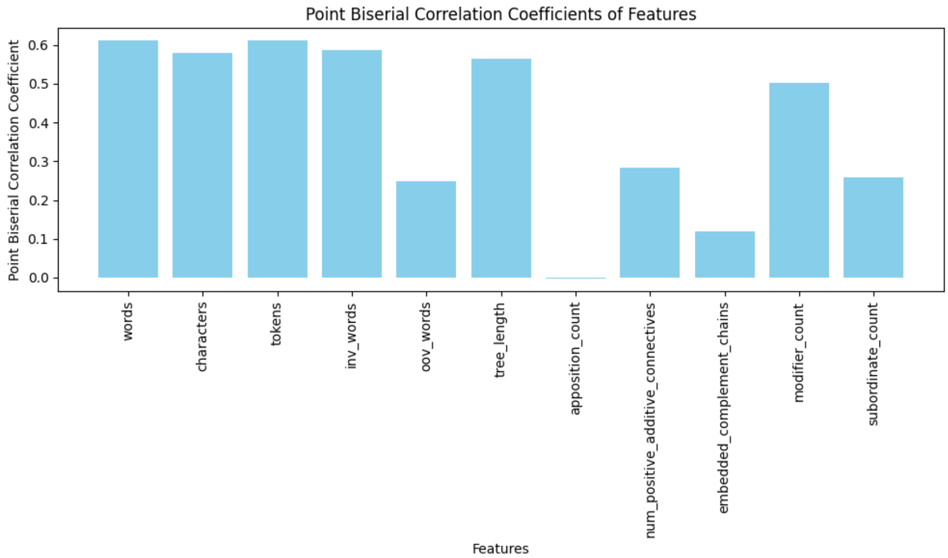

Figure 21: Biserial correlation for the features for GPT

5.8 SVM classifier

Cross validation : Using GridSearchCV for validation to find best parameters and with 5 folds and the parameters are: C =0.1, kernel = Linear. the model is configured to allow for some misclassification in the training data in exchange for better generalization to unseen data. The lower value of C (0.1) indicates that the model is less strict about misclassifications, resulting in a wider margin between the classes. This approach reduces the risk of overfitting, making the model more robust to variations in the data. The linear kernel assumes that the data is linearly separable, meaning the decision boundary is a straight line (or hyperplane in higher dimensions). The svm classifier for the new dataset reached to this accuracy:

	precision	**recall**	**f1-score**	**support**
0.0	0.83	0.94	0.88	1564
1.0	0.83	0.72	0.76	769
accuracy	0.83			
macro avg	0.83	0.78	0.79	2333
weighted avg	0.83	0.83	0.82	2333

Table 7: Classification Report

The f1-score difference between simple and complex sentences are now lower compare to the previous dataset. Without changing any features the accuracy reached to this: Model accuracy score with linear kernel : 0.8308 Model f1-score score with linear kernel : 0.7947 The other classifiers:

The other models are used for this classification to see which one is better.

5.9 Decision Tree classifier

For the decision tree classifier, RFE was used for feature selection, with 5-fold cross-validation and accuracy as the scoring metric, using entropy as the criterion. The exact same hyperparameters and features were applied to this dataset to evaluate the accuracy and F1-score of the classifier. This approach was taken to ensure a consistent comparison of results across different datasets, allowing for a clearer assessment of how well the model performs under the same conditions. By using identical settings, any variations in accuracy and F1-score can be attributed to differences in the datasets rather than changes in the model configuration.

5.10 Logistic regression classifier

For logistic regression in previous dataset, RFE was applied for feature selection, resulting in only one feature change: Embedded_complement_chains was removed, and number_of_subordinate was added to the features. The hyperparameters used for the classifier are the same for the GPT4o dataset. This method was chosen to maintain consistency when comparing results across different datasets, providing a more accurate evaluation of the model's performance under uniform conditions. By keeping the settings unchanged, any differences in accuracy and F1-score can be attributed to the datasets themselves, rather than modifications in the model's configuration.

5.11 Random Forest classifier

For the Random Forest classifier, feature importance was used to identify the most relevant features in previous dataset, and the only difference compared to the SVM model was the inclusion of number_of_subordinate. All other features remain the same. The hyperparameters used for the Random Forest classifier are the same as before.This strategy was adopted to ensure uniformity in the comparison of results across different datasets, allowing for a more accurate assessment of the model's performance under identical conditions. By maintaining the same settings, any changes in accuracy and F1-score can be attributed to differences in the datasets rather than alterations in the model's configuration.

5.12 Result

The SVM model with a linear kernel has the highest accuracy and F1-score among the models listed. SVMs are particularly effective in high-dimensional spaces and when the number of dimensions exceeds the number of samples. They are also memory efficient and work well with a clear margin of separation between classes. Decision Trees are prone to overfitting, especially with complex datasets. Even after tuning with cross-validation, the F1-score is lower, indicating that the model may not be as good at handling the balance between precision and recall compared to SVM. Logistic Regression is a good baseline model for binary classification, but it assumes a linear relationship between the features and the log-odds of the outcome. While it performs almost as well as the SVM in terms of accuracy, the F1-score is lower, suggesting that it may not handle imbalanced classes or non-linear relationships as effectively as SVM. Random Forests, being an ensemble of Decision Trees, tend to reduce the overfitting seen in individual trees. However, in this case, the performance metrics are the lowest among the listed models. This could be due to various reasons, such as the need for more hyper parameter tuning, the complexity of the dataset, or the random nature of the sampling process in Random Forests.

- High-Dimensional Spaces: SVMs are particularly effective in high-dimensional spaces, which could be beneficial if feature set is large.
- Margin Maximization: SVMs focus on maximizing the margin between classes, which can lead to better generalization on the test set.
- Robustness to Overfitting: While SVMs can overfit if not properly tuned, they generally handle overfitting better than Decision Trees.
- Effectiveness with Small Sample Sizes: SVMs can perform well even with smaller datasets, which might be advantageous depending on your dataset size.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
SVM	0.83	0.83	0.76	0.79
Decision Tree	0.80	0.65	0.72	0.68
Logistic Regression	0.82	0.81	0.75	0.70
Random Forest	0.78	0.65	0.70	0.63

Table 8: Comparison of Classification Models

5.13 Italian Dataset

Previous Study

In the previous study, seven features were used to classify simple and complex sentences. These features included characters, average length of dependencies, words not found in the dictionary, maximum tree depth, tokens, and subordinate clauses. The analysis was conducted using an Italian corpus, and the SVM classifier was applied to evaluate the performance of these features. Below are the results of the SVM classifier based on these selected features for the Italian corpus.

	Accuracy	**Recall**	**F1-score**	**Support**
Simple	0.89	0.84	0.84	716
Complex	0.70	0.81	0.75	402
Accuracy	0.81			
Macro Media	0.79	0.81	0.80	1118
Average Weighing	0.82	0.81	0.81	1118

Table 9: Classification Report

Now, the focus shifts to testing whether the features extracted from English can be applied effectively to another language, such as Italian. The 11 features identified from the English language are now being used for the Italian language, with adjustments made for language-specific differences. For example, the feature Modal verb – indicative mood is adapted as follows:
English modal verbs: Can, Could, Will, Would, Shall, Should, May, Might, Must, Ought, Could have, Would have, Should have, Might have.
Italian modal verbs: Dovere, Volere, Potere.
Similarly, other features have been modified based on the characteristics of the Italian language. The SVM classifier is once again used to classify simple and complex sentences, employing GridSearchCV with 5-fold cross-validation. The results for the 11 features, applied to the Italian dataset, are presented below. This is the Biserial correlation Coefficient for Italian language

Feature	**Value**
Words	0.61
Characters	0.62
Tokens	0.61
Inv_words	0.58
Oov_words	0.22
Tree_length	0.54
Apposition_count	0.022
Num_positive_additive_connectives	0.24
Embedded_complement_chains	0.15
Modifier_count	0.43
Subordinate_count	0.29

Table 10: Biserial correlation for the features

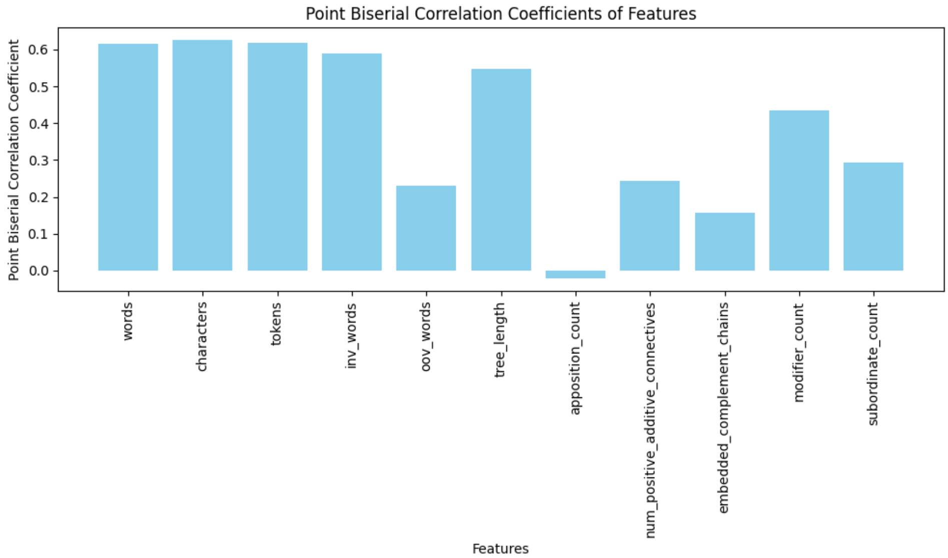

Figure 22: Biserial correlation for the features for italian dataset

5.14 SVM classifier

	precision	recall	f1-score	support
0	0.83	0.91	0.87	716
1	0.82	0.66	0.73	402
accuracy	0.82			
macro avg	0.82	0.79	0.80	1118
weighted avg	0.82	0.82	0.82	1118

Table 11: Classification Report

Cross validation : Using GridSearchCV for validation to find best parameters and with 5 folds and the parameters are: C =10, kernel = Linear. When performing cross-validation with a Support Vector Machine (SVM) using the parameters C = 10, and a linear kernel, the model is configured to prioritize minimizing classification errors on the training set. The high value of C (10) indicates a strict penalty for misclassification, which can lead to a model that may potentially overfit the training data, especially if the data contains noise or outliers. The linear kernel assumes that the data is linearly separable, meaning that the relationship between the features and the target variable can be captured by a straight line (or hyperplane in higher dimensions).

Compare to previous result it one percent better in accuracy and one for f1-score. The second classifier achieves a better balance between precision and recall for the simple class. With a precision of 0.83 and recall of 0.91 for the simple class, it effectively identifies most of the simple sentences while maintaining a high rate of correct identifications among those classified as simple. The F1-score for simple sentences in the second classifier is 0.87, which is higher than the 0.84 F1-score for simple sentences in the first classifier. This indicates that the second classifier is better at balancing precision and recall for simple sentences. The overall accuracy of both classifiers is similar (around 82%), but the second classifier maintains this accuracy with fewer data points. This suggests that the second classifier is robust and performs well even with a smaller dataset. The second classifier shows consistent performance across precision, recall, and F1-score metrics. The weighted average F1-score is 0.82, matching the overall accuracy, which implies that the model's performance is balanced across different metrics. If the primary goal is to accurately classify simple sentences (which might be more frequent or more critical in certain applications), the second classifier performs better. Its high recall (0.91) for simple sentences means it misses fewer simple sentences, and the high precision (0.83) means it has fewer false positives in this class.

the other models are used for this classification to see which one is better.

5.15 Decision Tree classifier

For the decision tree classifier, Recursive Feature Elimination (RFE) was employed for feature selection, utilizing 5-fold cross-validation with accuracy as the scoring metric and entropy as the criterion. The same hyperparameters and features were applied to this dataset to assess the classifier's accuracy and F1-score. This method was chosen to maintain consistency when comparing results across multiple datasets, enabling a more accurate evaluation of the model's performance under identical conditions. By keeping the settings unchanged, any observed differences in accuracy and F1-score can be attributed to variations in the datasets rather than adjustments to the model configuration.

5.16 Logistic regression classifier

For logistic regression in previous dataset, RFE was applied for feature selection, resulting in only one feature change: Embedded_complement_chains was removed, and number_of_subordinate was added to the features. The hyperparameters used for the classifier are the same for the Italian dataset. This method was chosen to maintain consistency when comparing results across different datasets, providing a more accurate evaluation of the model's performance under uniform conditions. By keeping the settings unchanged, any differences in accuracy and F1-score can be attributed to the datasets themselves, rather than modifications in the model's configuration.

5.17 Random Forest classifier

For the Random Forest classifier, feature importance was used to identify the most relevant features in previous dataset, and the only difference compared to the SVM model was the inclusion of number_of_subordinate. All other features remain the same. The hyperparameters used for the Random Forest classifier are the same as before.This strategy was adopted to ensure uniformity in the comparison of results across different datasets, allowing for a more accurate assessment of the model's performance under identical conditions. By maintaining the same settings, any changes in accuracy and F1-score can be attributed to differences in the datasets rather than alterations in the model's configuration.

5.18 Result

The SVM classifier stands out as the best-performing model among the four. It achieved the highest accuracy (82%) and F1-score (0.80), indicating that it is more effective in correctly classifying both simple and complex sentences.

The superior performance of the SVM classifier can be attributed to several factors. SVM works by finding the hyperplane that best separates the classes with the largest margin, which helps in achieving better generalization on unseen data. Additionally, SVM can use different kernel functions (linear, polynomial, RBF) to transform the data into a higher-dimensional space where it is easier to find a separating hyperplane. This ability makes it more adaptable to various types of data distributions. SVM is also particularly effective in high-dimensional spaces, which is often the case with text data after vectorisation. Furthermore, SVM is less prone to overfitting, especially in high-dimensional spaces, compared to other classifiers like Decision Trees, which can easily overfit the training data.

Overall, the Decision Tree and Random Forest classifiers, while intuitive and easy to interpret, did not perform as well as the SVM. Decision Trees tend to overfit the data, and while Random Forests mitigate this to some extent by averaging multiple trees, they still lag behind in performance. Logistic Regression provided decent performance but lacked the complexity to capture the nuances in the data as effectively as SVM. In conclusion, the SVM classifier's ability to find an optimal hyperplane, its robustness in high-dimensional spaces, and the use of kernel tricks contribute to its superior performance in distinguishing between simple and complex sentences.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
SVM	0.82	0.83	0.79	0.80
Decision Tree	0.81	0.70	0.73	0.74
Logistic Regression	0.81	0.81	0.75	0.72
Random Forest	0.76	0.73	0.70	0.65

Table 12: Comparison of Classification Models

5.19 Comparative Analysis of Classification Results Across Datasets

The overall comparison between the English dataset and the GPT-generated dataset reveals interesting insights into how the models perform across different types of data. While the accuracy for most models decreased slightly on the GPT dataset, the changes in precision, recall, and F1-score suggest that the models adapt differently to each dataset.

For the GPT dataset, there was a noticeable improvement in the F1-score for many models. This indicates that while the models may not have achieved the same level of accuracy as with the English dataset, they performed better at balancing precision and recall. This balance is particularly important in tasks like sentence classification, where it is crucial to minimize both false positives and false negatives.

The higher F1-scores on the GPT dataset suggest that the models were more consistent in their predictions when classifying complex and simple sentences, even though overall accuracy was lower. This might indicate that the GPT-generated dataset presents a slightly different distribution of sentence structures, requiring the models to better adjust their decision boundaries. In contrast, the English dataset seemed to allow the models to be more precise but slightly less balanced in terms of recall.

The findings highlight the importance of testing models across different datasets, as performance metrics such as accuracy, precision, and recall can vary based on the data's underlying structure and the specific features used. While accuracy alone may not always reflect the model's true performance, the F1-score improvements in the GPT dataset suggest that the models became better at handling nuanced cases of classification, which could make them more suitable for generalization to new datasets. Comparing the classification results between the Italian dataset for simple and complex sentences (shown in the latest table) and previous datasets (like the English and GPT-generated datasets), several differences and similarities are evident.

The accuracy for models applied to the Italian dataset is slightly lower compared to the English dataset but similar to the GPT dataset. This indicates that the Italian dataset introduces additional complexity that affects the models' accuracy. The Italian language's structural differences and unique sentence formations might have contributed to the slight drop in performance.

Across most models, precision on the Italian dataset is either comparable to or slightly higher than the GPT dataset but slightly lower than the English dataset. This means that the models are still relatively good at identifying relevant sentences but are slightly more prone to false positives in the Italian dataset compared to English. The linguistic characteristics of Italian, such as syntax and morphology, could explain these variations.

Recall for the Italian dataset is relatively stable across models and is generally comparable to the recall in the English and GPT datasets. This suggests that the models are relatively consistent in identifying complex and simple sentences in Italian, though there are some challenges in capturing all relevant instances. This could be due to the complexity of Italian sentence structures, where sentences often involve more subordination or intricacy compared to English.

F1-scores for the Italian dataset are slightly lower than for the English dataset, reflecting the models' struggles to balance precision and recall on this dataset. However, compared to the GPT-generated dataset, F1-scores in the Italian dataset remain competitive, indicating the models' adaptability to different languages. This drop in F1-score could be a result of the inherent differences in grammatical structure and complexity between Italian and English.

The models applied to the Italian dataset show a consistent performance pattern compared to previous datasets. While accuracy and precision are slightly lower in the Italian dataset, recall remains relatively stable, and F1-scores reflect the models' capacity to adapt to linguistic differences. This comparison highlights the importance of adjusting and testing models across languages, as language-specific characteristics can influence classification performance.

6) Conclusions and future work

The analysis and implementation of various machine learning classifiers for distinguishing between simple and complex sentences have yielded insightful findings. Among the classifiers evaluated, the Support Vector Machine (SVM) emerged as the most effective, achieving the highest accuracy and an F1-score. This superior performance can be attributed to SVM's ability to find the optimal hyperplane that best separates classes, its robustness in high-dimensional spaces, and the utilization of kernel functions that enhance its adaptability to various data distributions.

In contrast, other classifiers like Decision Trees and Random Forests, despite their intuitive and easy-to-interpret nature, did not perform as well. Decision Trees tend to overfit, while Random Forests, although mitigating this issue to some extent, still lagged behind SVM in performance. Logistic Regression, while providing decent results, lacked the complexity needed to capture the nuances in the data as effectively as SVM.

A significant part of the study was the application of these features and classifiers to an Italian language dataset, which included features such as the number of modal verbs, average length of dependencies, and various syntactic structures. The results were promising, with the SVM classifier demonstrating a slight improvement in accuracy and F1-score compared to the English dataset.

The findings emphasize the importance of selecting appropriate features and classifiers for specific language datasets. The use of grid search cross-validation in tuning the SVM parameters played a crucial role in optimizing the model's performance. Additionally, the robustness of the SVM classifier, even with a smaller dataset, highlights its potential for broader applications in natural language processing tasks.

Future work should focus on further refining these classifiers and exploring additional features that could enhance the classification accuracy. One promising direction is the integration of large language models (LLMs) such as BERT (Bidirectional Encoder Representations from Transformers). Researchers can train BERT on the given dataset to classify sentences as simple or complex. This approach strength BERT's deep contextual understanding of language, potentially leading to even more accurate classification.

Moreover, it would be beneficial to apply the extracted features from the current study to new datasets and analyze their performance using BERT. This step will determine whether the identified features generalize well across different datasets and languages, providing insights into their robustness and applicability. By comparing the performance of BERT-based models with traditional classifiers, researchers can better understand the strengths and limitations of various approaches.

Expanding the datasets to include more languages and varied sentence structures would provide a more comprehensive understanding of the classifiers' capabilities. Additionally, integrating advanced techniques such as deep learning models could potentially offer significant improvements in sentence classification tasks.

This study shows the effectiveness of the SVM classifier in handling the complexity of natural language and sets a strong foundation for future research in this domain. The continuous evolution of machine learning algorithms promises even greater advancements in text processing and classification, paving the way for more sophisticated and accurate language models. By exploring the potential of LLMs like BERT, future research can further enhance our ability to classify and simplify text, contributing to the accessibility of information for diverse audiences.

Credits

Thanks to Professor Claudio Sartori for supervising me and the PwC company for giving me this opportunity to do the project. And Special thanks to Mrs.Anna Elisabetta Ziri my manger in PwC.

Bibliography

[1] Chandrasekar et al. Motivations and methods for text simplification. 1996. URL: https://aclanthology.org/C96-2183.

[2] Daniel Jurafsky and James H.Martin. *Speech and Language Processing*. Pearson Prentice Hall,, 2009. URL: https://pages.ucsd.edu/ bakovic/compphon/Jurafsky,

[3] Heaps and H. S. *Information Retrieval: Computational and Theoretical Aspects*. McGraw Hill series in computer science. Academic Press, 1978. URL: https://dl.acm.org/doi/10.5555/539986.

[4] Berwick and Robert. *An Idiot's guide to Support vector machines*. URL: https://web.mit.edu/6.034/wwwbob/svm.pdf.

[5] Nivre and Joakim. An efficient algorithm for projective dependency parsing." in: Proceedings of the eighth international conference on parsing tech-nologies. 2003. URL: https://aclanthology.org/W03-3017/.

[6] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. 2005. URL: https://aclanthology.org/H05-1066/.

[7] Noam Chomsky. Three models for the description of language. 1956. URL: https://ieeexplore.ieee.org/document/1056813.

[8] Alfred V. Aho and Jeffrey D. Ullman. The theory of parsing, translation, and compiling. 1972. URL: https://dl.acm.org/doi/book/10.5555/578789.

[9] Joakim Nivre and Daniel Fernández-González. Squibs: Arc-eager parsing with the tree constraint. 2014. URL: https://aclanthology.org/J14-2002.

[10] Artur Kulmizev and Elena Fano Joakim Nivre Miryam de Lhoneux, Johannes Gontrum. Deep contextualized word embeddings in transition-based and graph-based dependency parsing. 2019. URL: https://aclanthology.org/D19-1277/.

[11] Matthew Honnibal, Yoav Goldberg, and Mark Johnson. A non-monotonic arc-eager transition system for dependency parsing. 2013. URL: https://aclanthology.org/W13-3518/.

[12] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. 2015. URL: https://aclanthology.org/D15-1162.

[13] Schmid H. Probabilistic part-of-speech tagging using decision trees. 1994. URL: https://citeseerx.ist.psu.edu/document repid=rep1type=pdfdoi=bd0bab6fc8cd43c0ce170ad2f4cb34181b31277d.

[14] Schmid H. Improvements in part-of-speech tagging with an application to german. 1999. URL: https://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/data/tree-tagger2.pdf.

[15] Andre Kempe. Probabilistic tagging with feature structures. 1994. URL: https://aclanthology.org/C94-1025/.

[16] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. 1988. URL: https://aclanthology.org/A88-1019/.

[17] J. P. Kincaid, Jr. Fishburne, Robert P., Rogers, Richard L., and Chissom Brad S. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. 1975. URL: https://apps.dtic.mil/sti/citations/ADA006655.

[18] Coleman, Meri Liau, and T. L. A computer readability formula designed for machine scoring. 1975. URL: https://psycnet.apa.org/record/1975-22007-001.

[19] David Kauchak and Teerapaun Tanprasert. Flesch-kincaid is not a text simplification evaluation metric. 2021. URL: https://aclanthology.org/2021.gem-1.1/.

[20] Dominique Brunato, Lorenzo De Mattei abd Felice Dell'Orletta, Benedetta Iavarone, and Giulia Venturi. Is this sentence difficult? do you agree? 2018. URL: https://aclanthology.org/D18-1289/.

[21] DOUGLAS BIBER and BETHANY GRAY. Grammatical change in the noun phrase: the influence of written language use. 2011. URL: https://www.cambridge.org/core/journals/english-language-and-linguistics/article/grammatical-change-in-the-noun-phrase-the-influence-of-written-language-use/AE0D25C8DE604BEB23602923DCC9C7B9.

[22] Hanan Khattab Omar Alsanawi. Apposition in english: A syntactic study in narrative and scientific texts. 2011. URL: https://www.researchgate.net/publication/332513870.

[23] Hanan Khattab Omar Alsanawi. Linguistic valency in grammar. 2023. URL: https://www.thoughtco.com/valency-grammar-1692484.

[24] Haeun Kim, Sungjin Koo, and Jungok Bae. Positive, negative, and nil effects of connectives in written stories: Analysis by proficiency groups. 2015. URL: isli.khu.ac.kr/journal/content/data/32_S/6.pdf.

[25] M Ryan Bochnak, Vera Hohaus, and Anne Mucha. Variation in tense and aspect, and the temporal interpretation of complement clauses. 2019. URL: https://academic.oup.com/jos/article/36/3/407/5532629.

[26] John Beavers and Andrew Koontz-Garboden. *The meaning of verbal roots and the roots of verbal meaning*. Oxford University, 2019. URL: https://personalpages.manchester.ac.uk/staff/andrewkg/root-semantics-book.pdf.

[27] Dominique Brunato, Lorenzo De Mattei, Felice Dell'Orletta, Benedetta Iavarone, and Giulia Venturi. Is this sentence difficult? do you agree? 2018. URL: https://aclanthology.org/D18-1289/.

[28] F. R. Palmer. *Mood and Modality*. Cambridge University Press, 2012 URL: https://www.cambridge.org/core/books/mood-and-modality/70959BB4A71A4C305A0A0695634A91EC.

[29] louisville.edu. Modifiers. 2012. URL: https://louisville.edu/writingcenter/for-students-1/handouts-and-resources/handouts-1/modifiers.

[30] Emmanuel Solorzano. Subordinating conjunctions and subordinate clauses. 2018. URL: https://pressbooks.pub/unmcorewriting/.

[31] Felice Dell'Orletta, Simonetta Montemagni, and Giulia Venturi. Read–it: Assessing readability of italian texts with a view to text simplification. 2011. URL: https://aclanthology.org/W11-2308/.

[32] Italian natural language processing lab. 2018. URL: http://www.italianlp.it/.

[33] *Wikipedia Dataset*. 2019. URL: https://www.kaggle.com/datasets/ankurzing/sentiment-analysis-for-financial-news?select=all-data.csv.

[34] *wikisentences dataset.* 2019. URL: https://www.kaggle.com/datasets.

[35] Zenodo. *Webiss Dataset.* 2019. URL: https://zenodo.org/records/205950.

[36] *Kaggle Financial-Dataset.* URL: https://www.kaggle.com/datasets.

[37] *Huggingface Dataset.* 2019. URL: https://huggingface.co/datasets/sentence-transformers/wikipedia-en-sentences.

[38] Prompt. 2024. URL: https://platform.openai.com/docs/guides/prompt-engineering.

[39] Khalid Iqbal and Muhammad Shehrayar Khan. Email classification analysis using machine learning techniques. 2022. URL: https://www.researchgate.net/publication/360474855.