KlarTextCoders at StaGE: Automatic Statement Annotations for *German Easy Language*

Akhilesh Kakolu Ramarao[1], Wiebke Petersen[1], Anna Sophia Stein[1], Emma Stein[2], Hanxin Xia[1]
Heinrich-Heine-Universität[1], Georg-August-Universität[2]
{akhilesh.kakolu.ramarao,wiebke.petersen,anna.stein,hanxin.xia}@uni-duesseldorf.de,
emma.stein@stud.uni-goettingen.de

Abstract

This paper presents our approach to the GermEval 2024 task, "Statement Segmentation in German Easy Language (StaGE)," addressing both subtasks: predicting the number of statements and identifying statement spans. We introduce a novel method integrating part-of-speech information with pre-trained BERT models, achieving leading performance in both the subtasks. For the statement count prediction (subtask 1), our model achieved a precision of 0.65, a recall of 0.68, and an F1-score of 0.65, with a Mean Absolute Error (MAE) of 0.36 and Mean Squared Error (MSE) of 0.43. For statement span annotation (subtask 2), we adapted our BERT model (used for subtask 1) to perform token-level classification, achieving a chrF score of 0.36 and a Jaccard similarity of 0.29. We also detail our exploration of alternative approaches to the shared task, including a rule-based system, LLMs, and traditional machine learning models. These machine learning models used a comprehensive feature set, combining Abstract Meaning Representation (AMR) features to capture deep semantic structures, part-of-speech (POS) tags for syntactic information, and other linguistic features.

Keywords: Automatic Text Simplification, Leichte Sprache, Statement Segmentation, German Easy Language.

1 Introduction

Easy Language is a simplified linguistic form that excludes complex grammatical and lexical features (Maaß and Bredel, 2017). Using German Easy Language, or: *Leichte Sprache* named in original German concepts, offers substantial advantages to a diverse range of users. A recent LEO study found that 6.2 million Germans have a low literacy level and struggle with reading simple sentences (Buddeberg and Grotlüschen, 2020). This includes, for example, individuals who are not native speakers of German or those with learning disabilities. However, other groups may also benefit from Easy Language: educators and therapists may find these adaptations beneficial in making educational materials more accessible and engaging. Furthermore, governmental entities sharing complex legal or bureaucratic information can leverage these tools to ensure their communications are comprehensible to a broader audience, potentially enhancing civic participation and adherence to regulations. Since 2011, German regulatory frameworks have mandated that governance bodies ensure their online content conforms to higher accessibility standards (Bundesministerium des Innern und für Heimat, 2011). This includes mandatory German Easy Language information on the main contents of the website, notes on navigating the website, and further information available in German Sign Language and in Easy Language.

Developing an algorithm capable of annotating statements in Easy Language could significantly streamline the process of composing and reviewing such texts, simplifying the publication of German Easy Language materials. Additionally, this technology could increase the quality of these texts by providing authors with deeper insights into the prevalence of specific statements within a text.

To foster this development, the "Statement Segmentation in German Easy Language (StaGE)" workshop from this year's GermEval task is comprised of two sub-tasks. The first task (subtask 1) involves predicting the number of statements for the given dataset, while the second task (subtask 2) asks to predict the position of the statement spans. The data for this task consists of Easy Language sentences sampled from Hurraki[1], an encyclopedia similar to Wikipedia for German Easy Language. For subtask 1, our submitted model achieved a 0.35 mean accuracy error (precision: 0.66, recall: 0.68,

[1] https://hurraki.de/wiki/Hauptseite

f1-score: 0.66) on the provided evaluation set. For the subtask 2, our submitted model achieved a chrF score of 0.36 and a Jaccard similarity of 0.29.

Our main contributions are: 1) We presented a multi-faceted approach to the shared task, using rule-based parser, machine learning classifiers with various linguistic features, Large Language Models (LLMs) and fine-tuning pretrained language model. 2) Integration of part-of-speech (POS) information with pre-trained language model for both subtasks. 3) We approached the statement span annotation task (subtask 2) as a token-level classification problem. This allowed the model to leverage transfer learning from the broader statement-level classification task (subtask 1).

All the code and data are available on GitHub[2].

2 Background

Text simplification has been an ongoing task attracting joint efforts from researchers of various disciplines across the globe (Shardlow, 2014). Although initially conceptualized as a method reducing the processing load of NLP systems by reducing text complexity (Chandrasekar and Srinivas, 1997), the application areas of Easy Language have expanded to assisting individuals with low literacy and reading comprehension difficulties, enhancing L2 language acquisition and facilitating the integration of migrants (Al-Thanyyan and Azmi, 2021; Steinmetz and Harbusch, 2022; Bock, 2014).

In the German context, governmental mandates aimed at improving public accessibility have led to the widespread availability of German Easy Language texts, known as *Leichte Sprache*, on the Internet (Ebling et al., 2022; Asghari et al., 2023). This has also led to the creation of text-based corpora such as, the document-level aligned corpus created by Gonzales et al. (2021), which contains full articles paired with simplified summaries collected from Swiss news magazine *20 Minuten*, and sentence-level aligned Simple German dataset curated by Toborek et al. (2023).

Early approaches to automatic text simplification in German focused on rule-based methods, such as the system developed by Suter et al. (2016), which primarily used syntactic transformation rules. Subsequent studies introduced discourse-based techniques (Niklaus et al., 2019) and lexical simplification strategies (Siegel et al., 2019).

More recent research viewed text simplification as a sequence-to-sequence task, where the input is a complex text and the output is a simplified version of the same text. For instance, Säuberli et al. (2020); Spring et al. (2021); Ebling et al. (2022) used transformer-based (Vaswani et al., 2017) sequence-to-sequence models. Gonzales et al. (2021) further extended this approach by fine-tuning mBART (Liu et al., 2020) model. Additionally, prompting techniques have been explored by Ryan et al. (2023) and Alva-Manchego et al. (2021), while Mallinson et al. (2020) used few-shot and zero-shot learning using BLOOM model (Scao et al., 2023). A thorough evaluation of the neural approaches is presented in Stodden (2024), providing a comprehensive comparison of their performance.

Despite these advancements, the automatic text simplification for German texts remains a persitent challenge (Schomacker et al., 2023). A major challenge in automatic text simplification is accurately identifying individual statements within complex sentences, a crucial step in improving the effectiveness of current simplification methods.

3 Task

The following section briefly outlines the "Statement Segmentation in German Easy Language (StaGE)" task at GermEval 2024 (Schomacker et al., 2024).

3.1 Annotation guidelines

A sentence in German Easy Language differs from regular language in a couple of ways and should adhere to its own set of rules. Ideally, a sentence in *Leichte Sprache* contains three arguments: subject, object, and verb. Additional information will be regarded as extra statements. The provided annotation guidelines[3] follow the recommendations drafted in the DIN SPEC 33429[4].

3.2 Dataset

For the task, four datasets are provided by the organizers, three reserved for development (train, trial, and test) and one for evaluation (eval). The trial set is the smallest split with 26 sentences, followed by test with 416, and finally, training with 1,530 sentences. The number-of-statement counts for each

[2] https://github.com/ansost/easy-to-read

[3] https://german-easy-to-read.github.io/statements/annotations/

[4] https://www.dinmedia.de/de/technische-regel-entwurf/din-spec-33429/364785446. Last accessed: 2nd August, 2024.

split can be seen in Table 1.

The small trial set was available at the start of the task for a general overview. Train data was then published for initial model development. Then the test set was released to enhance the development. Finally, eval was made public for blind model evaluation and scoring.

The train set has a token count of 5,093 with an average sentence length of 7.59. The token count and average sentence length of the test set are 1,142 and 7.35. Sentences in the eval set contain 7.54 tokens on average.

n. statements	trial	train	test	eval
0	1	449	-	-
1	15	1530	258	437
2	9	732	123	332
3	1	191	28	89
4	-	38	6	14
5	-	3	1	5
6	-	1	-	1

Table 1: Distribution of number of statements in trial, test and train.

4 Methodology

In this section, we describe different methods used in our study. We begin by explaining the Rule-based parser (Section 4.1), features used for machine learning classifiers (Section 4.2), BERT (Section 4.3) and finally the Large Language Models (LLMs) (Section 4.4).

4.1 Rule-based

We built a rule-based parser based on the annotation guidelines provided by the organizers. We used the German language model from SpaCy's *de_dep_news_trf*[5], which was trained on a dataset of German news articles. We began by performing a token-level analysis, where we iterated over each token in the sentence. For each token, we extracted the part-of-speech tag, dependency label, and morphological features using the parser mentioned above. Tokens in parentheses were counted as separate statements. Adjectives with noun heads were also counted, and they functioned as separate statements. Conversely, quantifiers, filler words, comparatives, and superlatives did not constitute independent statements.

In the next step, we checked for the existence of propositional phrases and their positioning in a sentence to see whether they added a new statement or not. For example, if a propositional phrase was part of a larger composite phrase, we counted it as one statement instead of multiple separate statements. Similarly, trivial propositions were not counted as separate statements. A sentence with date and year was treated as a distinct statement.

We then grouped tokens into clauses based on their dependencies and part-of-speech tags. We analyzed each clause to determine if it contained a subject, verb, or object. The clauses that met this criteria were treated as separate statements.

4.2 Feature-based

In order to experiment with classical feature-based machine learning approaches, a set of features has been extracted.

4.2.1 Feature extraction

The features were chosen to cover a spectrum of linguistic characteristics as broad and diverse as possible. This includes features extracted from syntax trees, features of meaning representations, simple length features and embedding vectors contextualized by a LLM. Some features have been inspired by our previous work on judging text simplicity (Arps et al., 2022) and on machine generated text detection (Ciccarelli et al., 2024). Altogether, we extracted 3,968 features, of which we kept only those 1,310. which applied to at least five items from the union of the train and the trial data.

Dependency tree features We parsed the phrases using the dependency tree parser with Spacy's *de_core_news_sm*[6] language model (the small model was used due to computational limitations). For each dependency tree, we have extracted all maximal paths, starting from the root node. From these, we extracted as features: 1) all dependency sequences of the maximal paths and their prefixes and 2) all pairings of maximal dependency paths with the POS tag of their nodes. Figure 1 (appendix) illustrates the extracted features. Furthermore, we included some more general features like minimal/maximal/average length of dependency chains, number of roots and number of root childs. All this resulted in 2,146 dependency tree features (of which 388 were kept as they apply to at least five items in train and trial).

[5] https://spacy.io/models/de

[6] https://spacy.io/models/de

AMR features To capture semantic features as well, we decided to use abstract meaning representations (AMRs) as a further source of features. AMR is a semantic representation framework that abstracts the meaning of a text by focusing on predicate-argument structure rather than surface form. It was first introduced in Langkilde and Knight (1998), and due to strict format specifications, it became one of the most influential representation formats for semantic parsing (Banarescu et al., 2013). Due to availability and performance, we have opted for the *amrlib* library for English.[7] Therefore we translated all texts to English using *GoogleTranslator* from the *deep-translator* library[8]. The translated texts were then parsed into AMRs. Similar to the dependency trees, we traversed the AMR, starting from the root node and collecting attribute paths as features. The main difference is that AMRs are not trees but general acyclic graphs. The following features have been collected (see Figure 2 in Appendix A for illustration): a) all attribute sequences starting at the root; b) each maximal path additionally paired with the instance types of its leave. Altogether this resulted in 1,776 AMR-features (888 apply to at least five items in train and trial).

Additional linguistic features Furthermore, we collected additional linguistic features. These features include simple counting features like number of tokens, mean number of characters per word, or number of punctuation counts and counts of important part-of-speech tags obtained via Spacy verb|de_dep_news_trf|; Spacy ver. 3.7.5;[9] Honnibal and Montani (2020).

For further syntactic features, we used the Berkeley Neural Parser (Kitaev et al., 2019; Kitaev and Klein, 2018)[10]. The direct results returned by the parser are NLTK tree forms (see Figure 3).

Based on the tree form of each sentence we extracted multiple features based on pre-defined rules for training the classifiers: 1) For NPs that contain more than two words (not only article and noun root), we count each occurrence within the sentence, so if a sentence has five such NPs, we record the big NP count as 5; 2) For Prepositional Phrases (PPs) that contain more than three words, we count each occurrence within the sentence, so if a sentence has two such PPs, we record the big PP count as 2; 3) If neither "(S" nor "(V" is present in the tree form, the current instance is not a sentence, i.e., 0-statement;.

Additionally, two readability formulas defined in the *textstat* library, namely Flesch and of the Wiener Sachtextformel (variant 1), have been applied to the phrases[11]. These formulas try to estimate how easy a text is to read on the basis of words per sentence and syllable or character per word counts. Both measures are specially designed to estimate the readability of German texts.

We also used a language complexity classification pipeline, which classifies texts into four language complexity classes and assigns them a score. Only the score was used as a feature [12].

Finally, the predictions on the number of statements made by the rule-based account as described in Section 4.1 has been added as an additional feature as well. Altogether, this group of additional features consists of 32 features (25 are kept, as they apply to at least five items in train and trial).

BERT features The BERT model *bert-base-german-cased* has been used without any finetuning to extract the last hidden state of the CLS token that can be seen as an embedding of the phrase.[13] Uniform Manifold Approximation and Projection (UMAP) has been used to reduce the number of dimensions to 10. The resulting 10 dimensions were used as additional features.

4.2.2 Data augmentation

In order to account for the unbalanced training set, in which more than 60% of the data belongs to the class '1 statement', we have decided to augment the data by round-trip translation. Therefore, all items with more than one statement have been translated into Finnish and back using Google Translator, and all examples with more than two statements have additionally been translated into Mandarin and back. Finnish and Mandarin were selected to maximize contrast with German. Neither language belongs to the Indo-European language family, and they represent opposite ends of the morphological spectrum. While Chinese is an isolating language, Finnish is agglutinative, providing a stark contrast to the inflectional nature of German.

[7] https://github.com/bjascob/amrlib

[8] https://pypi.org/project/deep-translator/

[9] https://spacy.io/models/de#de_dep_news_trf

[10] https://github.com/nikitakit/self-attentive-parser

[11] https://pypi.org/project/textstat/

[12] https://huggingface.co/krupper/text-complexity-classification

[13] https://huggingface.co/dbmdz/bert-base-german-cased

The idea is that this contrast in language structure reveals the underlying semantic content through roundtrip translations. It turned out that cases in which the re-translation matched the original were rare enough (for Finnish 6.6%, for Mandarin 3%) to be ignored. Thus, by the round-trip translation method new trainings data with differing feature values could be gained. We assumed that the number of statements is not affected by the translations.

4.3 BERT

We use BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) specifically *bert-base-german-cased*[14]. We further adapted this model by fine-tuning it for the subtasks, explained in Sections 5.1 and 5.2.

4.4 LLMs

Since Large Language Models are already finding various applications in the area of text simplification Tan et al. (2024); Baez and Saggion (2023), we wanted to explore how LLMs perform for annotating statements in simplified language.

We used *LLaMA-3-70B-Instruct* [15] for our analysis. To design the prompt, we started out with a simple one-shot prompt: "How many statements does this sentence have?". This prompt was then iteratively improved. One of the first changes we made was adding few-shot prompting and providing the model with example annotations from the training data in the prompt.

We also tested multiple ways of providing the annotation guidelines and specific rules from those annotation guidelines to the model. Automatic chain of thought prompting was also added since it has been shown to improve LLM performance in the past (Wei et al., 2022; Zhang et al., 2022). Initially, the prompts were given over *Huggingchat*[16], but an API provided faster and more reproducible results. This way, it allowed us to provide one sentence at a time instead of chunks of the dataset, which also improved performance. We also incorporated both system and user-level prompts and instructed the LLM to return the data in a specific format for easier analysis of the results. We used *LLaMA* model through an API endpoint provided by the KISSKI AI Service Centre for Sensitive and Critical Infrastructures[17]. All queries were made using a seed to ensure reproducibility. Models, scripts, prompts, and hyperparameters can be found on our GitHub.

[14] https://huggingface.co/google-bert/bert-base-german-cased

[15] https://huggingface.co/meta-LLaMA/Meta-LLaMA-3-70B-Instruct

[16] https://huggingface.co/chat/

5 Experiments

In this section, we provide a detailed evaluation of our approach to the shared task, specifying how we built the rule-based parser, different machine learning classifiers, and used Large Language Models (LLMs) (mentioned in Section 4).

5.1 Subtask 1: Determining the number of statements

The performance results of our models tackling subtask 1, as described in Section 4, are stated in Table 2 and compared to a most frequent class dummy classifier (MFC, model 0).

Nr	Model	Data	MAE	Acc
Most frequent class dummy classifier				
0	MFC	ttt	0.655	0.499
Rule-based parser				
1	RuleParser	tt	0.891	0.336
Feature-based (baselines)				
2	MLP+allFeat	ttt	0.426	0.622
3	SVM+allFeat	ttt	0.449	0.618
4	LR+allFeat	ttt	0.376	0.663
5	RF+allFeat	ttt	0.377	0.67
Feature-based (feature subsets + augmented data)				
6	LR+allFeat	ttt+au	0.367	0.67
7	RF+allFeat	ttt+au	0.354	0.688
8	RF+BERTFeat	ttt	0.625	0.498
9	RF+AMRFeat	ttt	0.559	0.544
10	RF+DepTreeFeat	ttt	0.419	0.634
11	RF+addFeat	ttt	0.39	0.646
12	LR-75-per-type	ttt+au	0.331	0.698
13	LR-200-overall	ttt+au	0.328	0.698
LLMs				
14*	LLMPrompting	tt	−−	0.668
15	**BERT+POStags**	**90%ttt**	**0.36**	**0.689**

Table 2: Performance comparison of different models tested on the eval data for subtask 1 (models marked by $*$ are trained and tested only on the original data sets). Training data differs (ttt: original train, trial and test data, ttt+au: ttt plus augmented data as described in Section 4.2.2). The feature-based approaches have been trained on all (allFeat) or a subset of the features as described in Section 4.2.

Rule-based parser Our initial approach to the task involved developing a rule-based parser that followed the annotation guidelines provided by the organizers (refer Section 4.1). The parser was then trained on the provided training dataset of 2,989 samples. It achieved an accuracy of 57% on the test dataset (consisting of 417 samples).

[17] https://kisski.gwdg.de/

Feature-based classifiers We tested four different machine learning classifiers implemented in Scikit-Learn (Pedregosa et al., 2011): Random Forest (RF), Support-Vector-Machine (SVM), Multilayer-Perceptron (MLP) and a logistic regression (LR) model (see Table 2, model 2-5). All classifier parameters have been set to the default values (for the MLP one hidden layer of size 128 was chosen).

Each classifier was trained on the combined train, trial and test set and evaluated against the eval set. Zero-statements were removed from all splits. In the training all features described in Section 4.2 have been used if they were attested for at least 5 phrases in the train and trial set. Table 2 shows that all classifiers beat the dummy baseline (MFC) and that RF and LR perform best of the baselines by outperforming MFC by 0.17 for accuracy (Acc.) and 0.28 for mean absolute error (MAE).

As on the original train and test data provided by the organizers, RF slightly outperformed LR all further experiments have been done with RF. Augmentating the training data did improve the results only slightly (compare model 7 to 5 and model 6 to 4 in Table 2). Additionally, we tested how different subsets of the feature set influence the performance (models 8-11 in Table 2). The additional linguistic features ('Add.') performed best, followed closely by the dependency tree features (DepTreeFeat). Significantly worse are the AMR and the BERT features (in this order).

To get a better grip on our features we have extracted from a Random Forest classifier the importance rankings for the features. Tables 3, 4 and 5 in Appendix show the 10 most important features for the first three classes. The 20 most important features over all classes can be seen in Table 6. It turns out that length features (length of the AMR representation, length of the maximal dependency chain, number of tokens, ...) are the most important features for classification. That was to be expected as longer sentences tend to include more statements. Interestingly, although a classifier trained solely on the BERT features (Table 2, model 7) does not perform very well, all 10 BERT features (UMAP0 up to UMAP9) belong to the overall top 20 features (see Table 6).

Based on the feature importance scores we trained LR models on feature subsets. As data augmentation lead at least to a slight improvement, we included the augmented data into our training data. First, we selected for each feature type class the n most important features (if available). We tested for n between 5 and 150 of which n=75 turned out best (see model 12 in Table 2). Second, we considered all features at once and selected the n overall most important features testing for n between 5 and 1,000. Here, n=200 turned out best (see model 13 in Table 2).

Thus, for the LR model MAE can be reduced by data augmentation by 0.01 (compare model 6 to 4) and by feature selection by an additional 0.04 (compare model 13 to 6). The accuracy improved by 3.5 percentage points.

LLM prompting We also experimented with *LLaMA* for this task. We tested its performance for a) annotating the number of statements, b) predicting the statement spans, and c) identifying whether a specific rule in the annotation guidelines applied to a sentence. An example of this is to predict whether the adjectives in this sentence constitute a new statement or not. This approach was primarily exploratory, but we still want to share our results. An example prompt can be found in Appendix B.

Out of all of these applications, LLMs worked best for predicting the number of statements with an accuracy of 66.83% on the test set. However, this accuracy is mainly due to *LLaMA* over predicting sentences with one statement. *LLaMA* struggled with predicting statement spans, presumably because this is a much more complex task, where a mistake in the first reasoning step easily leads to errors down the line. To test the individual rules in the annotation guidelines, we crafted a set of sentences and manually annotated whether the specific rule we wanted to test was true or not. Due to the manual nature of this part, we tested on a smaller set of samples for the annotation rules, so the results should be viewed as more of an initial investigation. Out of all the rules, LLMs seemed to perform well at deciding whether an adjective adds a new statement or not and moderately well for predicting zero statements. For the other rules, its predictions were hardly better than chance. With a fine-tuned model and larger example sets, results might increase. However, given the lower explainability of LLMs compared to our other methods and since their accuracy did not match that of our other approaches, we decided not to pursue this method further.

5.1.1 Submission

We extend the BERT architecture (refer Section 4.3) to incorporate part-of-speech (POS) information alongside the contextual embeddings generated by BERT. This model consists of three main components: a pre-trained German BERT model, a POS encoder that transforms POS tags into dense representations, and a classifier that combines BERT outputs with encoded POS features. By encoding POS tags, the model potentially gains access to extra syntactic knowledge that might not be fully captured in BERT's learned representations alone.

The model's forward pass begins by processing the input text through the BERT model to get the contextual embeddings. We then extract the [CLS] token representation from BERT's last hidden state. After that, we get the POS tags for each input text using the SpaCy *de_core_news_lg*[18] language model and encode the POS tags using a simple one-hot encoding. At the same time, the POS tags are encoded using a linear layer to create dense representations. The BERT [CLS] token representation and the encoded POS features are concatenated and passed through a final linear classifier to predict the number of statements. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 2e-5 and train the model for 100 epochs with a batch size of 16.

We combine the trial, train, and test datasets provided by the organizers (for details, see Section 3.2), remove zero-statement sentences, and get 2,937 samples. We split this dataset into 90% training and 10% test datasets by randomly shuffling the samples. In the end, the training set and test set consist of 2,643 and 294 samples, respectively. On this test set, the model achieved an accuracy of 80%.

On the evaluation dataset provided by the organizers, our model achieved a precision of 0.65, a recall of 0.68, and an F1-score of 0.65. In terms of prediction accuracy, the model achieved a Mean Absolute Error (MAE) of 0.36, indicating that, on average, our predictions deviate by approximately 0.36 units from the true values. The Mean Squared Error (MSE) is 0.43, which is slightly higher than MAE, but the difference (0.07) is quite low, which means our model is generally consistent and has very few large outliers in the predictions.

[18] https://spacy.io/models/de

5.2 Subtask 2: Annotating the statement spans

The goal of the task is to identify and annotate distinct statements within a sentence, particularly when a sentence contains multiple statements. We approach this sequence labeling problem as a token classification task. This approach allowed us to adapt the BERT model used for subtask 1 (see Section 5.1.1), which is already fine-tuned for statement-level classification, to perform token-level predictions. In this framework, we transform the original attention span labels into token-level classifications. We use the `tokenized phrase` provided in the dataset (see Section 3.2), and each token is assigned a label corresponding to its position within the statement span. For example, consider the `tokenized phrase`, ['Alcopop', 'ist', 'ein', 'süßes', 'Getränk.'] which has the attention span of [[0, 1, 4], [3]] and is converted to [1, 1, 0, 2, 1], since 'Alcopop', 'ist' and 'Getränk.' belongs to the first span (labeled as 1), 'ein' belongs to none of the spans (labeled as 0) and 'süßes' belongs to the second span (labeled as 2).

5.2.1 Submission

The fine-tuned BERT used for subtask 1 (refer Section 5.1.1) is further fine-tuned for this token classification task. This approach leverages transfer learning, allowing the model to build upon the knowledge gained from the broader statement-level classification to perform the more specific token-level classification. We modify the top layers of the already fine-tuned model by retaining the pre-trained BERT layers, adding a part-of-speech (POS) encoder to incorporate syntactic information, and implementing a new classification layer for token-level predictions.

In the forward pass, the model first processes the input (tokenized phrase) through BERT, obtaining the last hidden state. It then encodes the POS tags using the POS encoder. These two outputs are concatenated along the last dimension, combining contextual information from BERT with the POS information. Finally, this combined representation is passed through the classifier to produce logits for each token. The loss is computed on the flattened logits and labels, treating each token as an independent classification problem (padded tokens are ignored).

We then fine-tuned this adapted model on the evaluation dataset (provided by the organizers) following the same split as subtask 1 (90% train and

10% test split), but the sentences were shuffled. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 2e-5 and train the model for ten epochs with a batch size of 16.

Our model's performance was evaluated using two key metrics: chrF and Jaccard similarity. The chrF score indicates the model's ability to capture character-level n-gram similarities between the predicted and reference texts (Popović, 2015). The Jaccard similarity is a statistical measure used to gauge the similarity and diversity of sample sets. The model achieved a chrF score of 0.36 and a Jaccard similarity of 0.29 on the evaluation dataset provided by the organizers.

6 Conclusion and Future work

Our experiments focused on both detecting the number of statements and subsequently detecting the annotation spans of Simple German sentences, namely both subtasks 1 and 2 of the GermEval "Statement Segmentation in German Easy Language (StaGE)" shared task.

In subtask 1 of detecting the number of statements (Section 5.1), the submitted model that extended the BERT architecture by incorporating POS features scored a 0.65 precision, 0.68 recall, and 0.65 F1-score on the evaluation dataset. The Mean Absolute Error (MAE) and the Mean Squared Error (MSE) were 0.36 and 0.43, respectively. The relatively small difference between MAE and MSE suggests consistent performance without significant outliers.

For the Subtask 2 of statement span annotation (Section 5.1), we adapted the BERT model used for subtask 1 to perform token-level classification. This approach achieved a chrF score of 0.36 and a Jaccard similarity of 0.29. While these scores indicate room for improvement, they demonstrate the model's ability to identify and annotate distinct statements within a sentence.

These results provide encouraging evidence for the effectiveness of our approach, particularly using BERT with POS information for both subtasks. Moreover, our exploration of various methods, from rule-based systems to fine-tuning language models, allowed us to gain insights into the strengths and limitations of different approaches for both subtasks.

6.1 Future work

Based on the results and approach described, we would like to propose some potential directions for future work.

We would like to investigate a multi-task learning approach, where the model is trained simultaneously on both statement identification and span annotation tasks, which could provide an improved performance across both tasks by using shared linguistic knowledge.

We plan to compare the performance of our BERT architecture with other pre-trained language models like RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019) or T5 (Raffel et al., 2020) or mBART (Liu et al., 2020). This will help us identify the most effective base model for the given tasks. With regard to the *LLaMA* approach, it would be interesting to explore fine-tuned models and integrate the LLM's judgment with other approaches, for example, rule-based.

Limitations

We acknowledge a few limitations in our approach. A significant limitation is that we did not perform hyperparameter tuning for our extended BERT model for both tasks. We focused on extending BERT but did not explore other pre-trained models. We did not conduct any analysis on how the POS information influences the model's decision. We did not conduct ablation studies to understand the individual contribution of different components of our model, such as the POS encoder or specific layers of our BERT architecture.

Ethics Statement

We fully complied with all rules, guidelines, and data usage policies set forth by the shared task organizers. We accurately represented our models' capabilities and limitations. *LLaMA* was used solely for inference purposes; we did not fine-tune our model. We aimed to use computational resources efficiently by leveraging pre-trained models. Our submissions reflect our own work.

CRediT authorship contribution statement

We follow the CRediT taxonomy[19]. Conceptualization: [AKR, ES, AS, HX, WP]; Formal Analysis: [AKR, ES, AS, HX, WP]; Investigation: [AKR, ES, AS, HX, WP]; Methodology: [AKR, AS, WP];

[19] https://credit.niso.org/

Supervision: [AKR, AS, WP]; and Writing – original draft: [AKR, ES, AS, HX, WP] and Writing – review & editing: [AKR, ES, AS, HX, WP].

Acknowledgments

We thank the organizers of both subtasks for their effort and for their help during the training and evaluation phases. We would like to thank Nele Mastracchio for helping in the initial ideation phase of this project. We acknowledge the use of Google Colab[20] for providing some of the computational resources necessary for this research.

References

Suha S Al-Thanyyan and Aqil M Azmi. 2021. Automated text simplification: a survey. *ACM Computing Surveys (CSUR)*, 54(2):1–36.

Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. 2021. The (Un)Suitability of Automatic Evaluation Metrics for Text Simplification. *Computational Linguistics*, 47(4):861–889.

David Arps, Jan Kels, Florian Krämer, Yunus Renz, Regina Stodden, and Wiebke Petersen. 2022. HHUplexity at text complexity DE challenge 2022. In *Proceedings of the GermEval 2022 Workshop on Text Complexity Assessment of German Text*, pages 27–32, Potsdam, Germany. Association for Computational Linguistics.

Hadi Asghari, Freya Hewett, and Theresa Züger. 2023. On the prevalence of leichte sprache on the german web. In *Proceedings of the 15th ACM Web Science Conference 2023*, pages 147–152.

Anthony Baez and Horacio Saggion. 2023. LSLlama: Fine-Tuned LLaMA for Lexical Simplification. In *Proceedings of the Second Workshop on Text Simplification, Accessibility and Readability*, pages 102–108, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *LAW@ACL*.

Bettina M Bock. 2014. „leichte sprache “: Abgrenzung, beschreibung und problemstellungen aus sicht der linguistik. *Sprache barrierefrei gestalten. Perspektiven aus der Angewandten Linguistik*, pages 17–51.

Klaus Buddeberg and Anke Grotlüschen, editors. 2020. *LEO 2018: Leben mit geringer Literalität*. wbv Publikation, Bielefeld.

Bundesministerium des Innern und für Heimat. 2011. Barrierefreie-informationstechnik-verordnung 2.0. https://www.barrierefreiheit-dienstekonsolidierung.bund.de/Webs/PB/DE/gesetze-und-richtlinien/bitv2-0/bitv2-0-artikel.html. accessed: 17.04.2024.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183–190.

Vittorio Ciccarelli, Cornelia Genz, Nele Mastracchio, Wiebke Petersen, Anna Stein, and Hanxin Xia. 2024. Team art-nat-HHU at SemEval-2024 task 8: Stylistically informed fusion model for MGT-detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1690–1697, Mexico City, Mexico. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sarah Ebling, Alessia Battisti, Marek Kostrzewa, Dominik Pfütze, Annette Rios, Andreas Säuberli, and Nicolas Spring. 2022. Automatic text simplification for german. *Frontiers in Communication*, 7:706718.

Annette Rios Gonzales, Nicolas Spring, Tannon Kew, Marek Kostrzewa, Andreas Säuberli, Mathias Müller, and Sarah Ebling. 2021. A new dataset and efficient baselines for document-level text simplification in german. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pages 152–161.

Matthew Honnibal and Ines Montani. 2020. spacy: Industrial-strength natural language processing in python.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

[20] https://colab.research.google.com/

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Christiane Maaß and Ursula Bredel. 2017. *Ratgeber Leichte Sprache: Die wichtigsten Regeln und Empfehlungen für die Praxis*. Duden. ISBN: 9783411912360.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2020. Zero-shot crosslingual sentence simplification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5109–5126, Online. Association for Computational Linguistics.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2019. DisSim: A discourse-aware syntactic text simplification framework for English and German. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 504–507, Tokyo, Japan. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830. Ver. 1.5.0.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Michael Ryan, Tarek Naous, and Wei Xu. 2023. Revisiting non-English text simplification: A unified multilingual benchmark. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4898–4927, Toronto, Canada. Association for Computational Linguistics.

Andreas Säuberli, Sarah Ebling, and Martin Volk. 2020. Benchmarking data-driven automatic text simplification for German. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI)*, pages 41–48, Marseille, France. European Language Resources Association.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina Mcmillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco de Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro von Werra, Leon Weber, Long Phan, Loubna Ben Allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-Shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri,

Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh Hajihosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael Mckenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel de Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-Aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2023. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. Working paper or preprint.

Thorben Schomacker, Miriam Anschütz, Regina Stodden, Marina Tropmann-Frick, and Georg Groh. 2024. Overview of the germeval 2024 shared task on statement segmentation in german easy language (stage). In *Proceedings of the GermEval 2024 Shared Task on Statement Segmentation in German Easy Language (StaGE)*, Vienna, Austria. Association for Computational Linguistics.

Thorben Schomacker, Tillmann Dönicke, and Marina Tropmann-Frick. 2023. Exploring automatic text simplification of German narrative documents. In *Proceedings of the 19th Conference on Natural Language Processing (KONVENS 2023)*, pages 139–148, Ingolstadt, Germany. Association for Computational Lingustics.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Melanie Siegel, Dorothee Beermann, and Lars Hellan. 2019. Aspects of linguistic complexity: A german – norwegian approach to the creation of resources for easy-to-understand language. In *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–3.

Nicolas Spring, Annette Rios, and Sarah Ebling. 2021. Exploring German multi-level text simplification. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1339–1349, Held Online. INCOMA Ltd.

Ina Steinmetz and Karin Harbusch. 2022. A text-writing system for easy-to-read German evaluated with low-literate users with cognitive impairment. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 27–38, Dublin, Ireland. Association for Computational Linguistics.

Regina Stodden. 2024. Reproduction of German text simplification systems. In *Proceedings of the Workshop on DeTermIt! Evaluating Text Difficulty in a Multilingual Context @ LREC-COLING 2024*, pages 1–15, Torino, Italia. ELRA and ICCL.

Julia Suter, Sarah Ebling, and Martin Volk. 2016. Rule-based automatic text simplification for german. In *Proceedings of the 13th Conference on Natural*

Language Processing, KONVENS 2016, Bochum, Germany, September 19-21, 2016, volume 16 of *Bochumer Linguistische Arbeitsberichte*.

Keren Tan, Kangyang Luo, Yunshi Lan, Zheng Yuan, and Jinlong Shu. 2024. An llm-enhanced adversarial editing system for lexical simplification. *Preprint*, arXiv:2402.14704.

Vanessa Toborek, Moritz Busch, Malte Boßert, Christian Bauckhage, and Pascal Welke. 2023. A new aligned simple German corpus. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11393–11412, Toronto, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic Chain of Thought Prompting in Large Language Models. *arXiv preprint*. ArXiv:2210.03493 [cs] version: 1.

Appendix

A Extracted Features

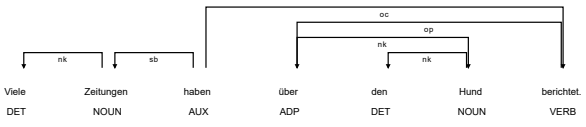

Figure 1: Dependency tree of phrase 'Viele Zeitungen haben über den Hund berichtet.' The following features are extracted from this tree: 'max_dep_length', 'mean_dep_length', 'min_dep_length', 'num_dep_paths', 'num_root', 'num_root_childs', 'oc#', 'oc#op#', 'oc#op#nk#', 'oc#op#nk#nk#', 'oc#op#nk#nk#DET', 'punct#', 'punct#PUNCT', 'sb#', 'sb#nk#', 'sb#nk#DET'

```
(r / report-01
      :ARG0 (n / newspaper
            :quant (m / many))
      :ARG1 (d / dog))
```

Figure 2: AMR of phrase 'Many newspapers have reported about the dog.' The following features are extracted from this AMR: 'arg0:', 'arg0:instance:', 'arg0:quant:', 'arg0:quant:instance:', 'arg0:quant:instance:many', 'arg1:', 'arg1:instance:', 'instance:'

```
((S
  (ADV Leider)
  (VVFIN bekomme)
  (PPER ich)
  (NP (PIAT keine) (NN Katze)))
($, ,))
```

Figure 3: An example of NLTK tree forms

B LLM prompts

The following is an example of an LLM prompt we used. Text in **bold** was not part of the prompt but is included here to illustrate the different prompt levels. For the sake of clarity, the prompt is slightly shortened, the full prompt can be found in our GitHub repository.

System prompt
You are an expert in German Easy Language.
User prompt
Give the statement spans of the sentence below.
For your decisions rely on the annotation guidelines provided below.
Provide your output in the form of a nested list. Return nothing but that list or the string "None" if the sentence only has one statement or zero statements.
Think step by step.
Three example sentences:
sentence: Eine sehr bekannte Alchemisten war Maria die Jüdin
statements: 1, statement spans: None;
(We provided two more examples but did not include them here for the sake of brevity.)

The sentence you should annotate is the following:
{sentence}
Annotation guidelines:
{annotation_guidelines}

C Feature Importance

Features	Importance
attr-arg1:attr-instance:	0.0088
amr_length	0.0084
attr-arg1:	0.0083
attr-arg2:	0.0077
attr-arg0:attr-instance:	0.0076
attr-arg2:attr-instance:	0.0075
attr-(mod):attr-instance:	0.0074
attr-(mod):	0.0074
attr-arg0:	0.0073
attr-time:attr-instance:	0.0071

Table 3: Ranked 10 most important AMR-features

Feature	Importance
max_dep_length	0.1012
mean_dep_length	0.0869
num_dep_paths	0.0787
num_root_childs	0.0523
sb#PRON	0.0137
sb#nk#DET	0.0134
sb#nk#	0.0123
mo#ADV	0.0108
num_root	0.0101
oa#	0.0089

Table 4: Ranked 10 most important dependency tree features

Feature	Importance
token_count	0.1738
mean_chars_per_word	0.0956
wiener_sachtextformel	0.0946
flesch_reading_ease	0.0814
num_compound	0.0725
num_nouns	0.0584
verb_count	0.0472
num_PP	0.0464
class_score	0.0395
num_S	0.0381

Table 5: Ranked 10 most important additional linguistic features

Feature	Importance
token_count	0.06
max_dep_length	0.0328
num_dep_paths	0.0328
mean_dep_length	0.0256
num_compound	0.0211
UMAP2	0.0179
UMAP4	0.0176
UMAP1	0.0176
UMAP0	0.0168
UMAP8	0.0166
UMAP9	0.0161
UMAP5	0.0161
UMAP7	0.016
verb_count	0.016
UMAP3	0.0158
flesch_reading_ease	0.0155
num_S	0.0154
wiener_sachtextformel	0.0153
UMAP6	0.0146
mean_chars_per_word	0.0146

Table 6: Ranked 20 most important features (over all feature classes)