RUPRECHT-KARLS-
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Institut für Computerlinguistik Heidelberg

Seminararbeit im Seminar
Language Technology for Education Assessment
Wintersemester 2017/2018

Language Level Analysis and Classification

Referent:
Dr. Magdalena Wolska

Verfasserin:
Julia Suter
Matrikelnummer 3348630

Masterstudiengang Computerlinguistik
5. Fachsemester

18. Januar 2018

Table of Contents

1 Introduction

Computers have become an important source of assistance in the educational sector, for both students and teachers alike. Even basic functions such as word processing are very helpful in writing and modifying texts, especially since spelling and grammar checkers give immediate feedback [Dikli, 2006]. Consequently, research toward educational applications has become a well-established field of NLP. Educational assessment addresses evaluating learners' abilities or knowledge, and includes tasks such as automatic essay scoring [Shermis and Burstein, 2003], [Burstein et al., 2003], short answer scoring [Leacock and Chodorow, 2003], [Nielsen et al., 2009] and test item generation [Heilman and Smith, 2010], [Becker et al., 2012].

In this report, I present a tool for identifying linguistic constructions and characteristics that allow inference about the writer's language skills. Experiments with a linear support vector classifier trained on these features demonstrate that this approach is fit for assigning language levels to unseen texts. Since research in educational assessment is often focused on English, I decided to implement this tool for German. To the best of my knowledge, my work is the first attempt at addressing this specific task, although it shows similarities to previous research on essay scoring and readability assessment [Flesch, 1948], [DuBay, 2004].

The first part of the project consisted of the selection of linguistic indicators and patterns that possibly could help in identifying the different language levels. Those features include information about part-of-speech, tempus, mode, use of subjunctive clauses, or passive voice. My tool returns information about the frequency of occurrence of these linguistic features and colors the identified features in the text for visualization. Both numeric information and the colored text may help in manually evaluating the language level of a given text.

In the second part of the project, I collected a small corpus of text samples for levels A1 to B2. I extracted the previously designed features and used them to train a linear support vector classifier for the four different language levels. This system clearly outperforms the baseline and reaches an accuracy of 89%. Feature agglomeration raises the performance to 96% accuracy.

1.1 Motivation

Language level classification differs from the more established essay scoring task. Although essay scoring systems usually include grammar and spelling correction or even measures for syntactic variety, they also focus on other aspects of essay writing, such as discourse analysis, lexical complexity, general style and content [Burstein et al., 2003]. My system pursues a more general goal and does not detect errors or analyze content or text structure. Therefore, it is not limited to a specific text form (e.g. essays) and remains domain-independent. Purely linguistic information that can be extracted with a dependency parser is used, which makes the system robust, stable, general and readily extensible.

The two parts of my work suggest two applications. In the first application, the linguistic information gained from the text can be used to generally analyze texts, or focus on specific patterns and structures, which can be detected, counted and highlighted in the text. This analysis assists manual evaluation of texts. If a specific linguistic phenomenon is in focus (e.g. passive voice), the system finds examples in texts, both when correcting a student's text and when searching for suitable teaching material or exercises. At the same time, the system can be used to determine which aspects of grammar are most prominent for a language level of any framework, which is valuable information when preparing for a test.

The second application, the actual language level classifier, can be used in two ways: it can assign a level to texts written by native speakers in order to examine whether they are appropriate reading material for a specific level, or it can be used to evaluate a student's writing level. For the latter case, however, no experiments were conducted in this work. The dataset used contains only texts written by native speakers (or in rare cases adapted and corrected students' answers). Since texts written by L2-learners often contain mistakes, the parser is expected to perform worse on them, which affects the whole feature extraction process and therefore accuracy. Although the main goal of this work is not to compute a German L2-learner's language level, it could certainly be adapted in future work to handle and possibly even identify common students' mistakes.

1.2 Applications for controlled natural languages

There is also a specific field that may benefit from both applications of my system: controlled natural languages. Controlled natural languages are languages that are restricted in grammar and dictionary in order to reduce complexity and ambiguity. They are either aimed at producing texts with enhanced readability for human readers, or focus on improving performance of natural language processing tasks [Kuhn, 2014]. The first application, especially paired with an interactive interface, could help in creating and automatically validating texts that are to be written in a controlled language. Forbidden structures (or vocabulary) could be highlighted to facilitate validation and corrections.

An example for a controlled natural language is easy-to-read language. It specializes on expressing information in a way that is understandable by all, including readers with low literacy skills. It is marked by reduced lexical and syntactic complexity. Guidelines define what structures are easy enough to understand, and which have to be paraphrased [Maaß and Bredel, 2016, p. 113]. A writer or translator of easy-to-read texts could therefore benefit from feedback on occurring linguistic structures.

Verified texts written in easy-to-read German are labeled with a quality seal and are sometimes divided into three levels: *A1*, *A2* and *B1*. They correspond to the *A1-B1* levels in the Common European Framework of Reference for Languages (CEFR) [Bock, 2014, p. 21]. The language level classifier could help in identifying texts of these levels to create a large collection of easy-to-read texts in German. Such a dataset would be key for the further promotion of easy-to-read language and to support research in this field, as there are very few easy-to-read language corpora available, and hardly any for German [Klaper et al., 2013].

Finally, both applications could be employed for the evaluation of automatic text simplification systems for German, as for example introduced in Suter et al. [2016], since quality assessment for text simplification is an active field of research [Štajner et al., 2016].

The language level is by far not the only text characteristic that could be captured by the classifier, however. Different domains, genres, and authors for instance will most likely show differences in the extracted features as well. In this work, I have not conducted any experiments in this direction but the classifier is expected to work for any set of classes with different distributions of the extracted features. Since the classifier seems to perform well even when trained on a relatively small dataset, experiments with a small number of samples are possible.

2 Dataset

In order to train a language level classifier, a collection of texts labeled with their language level is required. As I could not find a ready-to-use dataset, I manually searched the web for suitable samples. To avoid confusion and noise introduced by the use of different frameworks, I only considered texts that were clearly labeled with a level defined by the CEFR. Since I could not find a sufficient number of texts for the proficiency levels *C1* and *C2*, the collection is limited to texts for the beginner levels *A1* and *A2*, and the intermediate levels *B1* and *B2*. I did not differentiate between finer graduations, such as level *A1.1* and *A1.2*.

The texts were extracted from various websites offering free exercises and mock exams for L2-learners of German. Table 1 shows how many texts of which level were extracted from each source. The texts were stripped of titles and author markers but were otherwise left unmodified. I collected at least 60 samples (with 3 to 50 sentences) for each level, which constitutes a relatively small corpus. For further experiments, I recommend the assembly of a larger collection. The extraction of texts from different sources does not pose a problem as long as the CEFR level is clearly assigned. It may even help to avoid overfitting.

[1] `https://www.cornelsen.de/studio_21/1.c.3237778.de` Accessed: 13.08.2017
[2] `https://deutschsek.wikispaces.com/Leseverstehen+1` Accessed: 13.08.2017
[3] `https://german.net/reading/` Accessed: 13.08.2017
[4] `https://www.hueber.de/seite/pg_lehren_lesetexte_sri` Accessed: 13.08.2017
[5] `https://www.schubert-verlag.de/aufgaben/` Accessed: 13.08.2017
[6] `http://bfu.goethe.de/b1_mod/lesen.php` Accessed: 13.08.2017
[7] `https://deutschkollegstuttgart.wordpress.com/tag/deutschkurs-intensiv-a1/` Accessed: 13.08.2017
[8] `https://www.klett-sprachen.de/download/3680/Modelltest-Zertifikat-Deutsch-B12.pdf` Accessed: 13.08.2017
[9] `https://www.telc.net/fileadmin/user_upload/telc_deutsch_b2_uebungstest_1.pdf` Accessed: 13.08.2017
[10] `https://www.tsu.ge/data/file_db/faculty_medicine/ger-B%202%20-2009-Dokt.pdf` Accessed: 13.08.2017

Source	A1	A2	B1	B2	Total
Cornelsen[1]	13	19	33	8	73
DeutschSek[2]	4	0	1	1	6
GermanNet[3]	11	12	9	3	35
SchrittePlus[4]	21	15	0	0	36
Schubert Verlag[5]	1	2	0	0	3
Goethe-Zertifikat Modelltest[6]	5	0	3	16	24
Deutschkolleg Stuttgart[7]	7	14	19	9	49
Klett-Sprachen Modelltest[8]	0	0	9	0	9
TELC Language Test[9]	0	0	0	23	23
Tsu[10]	0	0	0	6	6
Total	**62**	**62**	**74**	**66**	**264**

Table 1: Number of language level samples extracted from each source website.

3 Features

Since the features are syntactically motivated, the text is parsed as a preprocessing step. To enable the direct processing of raw input text, I employed the hybrid dependency parser *ParZu* [Sennrich et al., 2009] and the associated rule-based incremental entity-mention system for coreference resolution *CorZu* [Klenner and Tuggener, 2011]. As opposed to many essay scoring systems, I did not include any lexical and text structure features to show that the language level can be determined by looking at syntactic features alone. In future work, my system could be extended with lexical and text structure-based features, though.

The features were partially inspired by the *Sprachprofilbogen*[11][12](language profile sheet) introduced in Griesshaber [2005]. Furthermore, I studied the grammar overviews for levels A1 to B1 and created feature groups, according to the level at which a construction or specific aspect of grammar is learned [Witzlinger, 2017], [Dengler and Sieber, 2017] (see Chapter 3.2).

3.1 Feature Description

The features described below represent the frequency of a specific characteristic, form, pattern or linguistic phenomenon in a given text. Unless stated otherwise, these frequencies are number of occurrences by the number of sentences in the text. The use of frequencies represents an inherent normalization step for text length.

[11] `https://spzwww.uni-muenster.de/griesha/sla/tst/profilbogens-0906.pdf` Accessed: 13.08.2017

[12] `https://spzwww.uni-muenster.de/griesha/sla/tst/profilbogeng-0906.pdf` Accessed: 13.08.2017

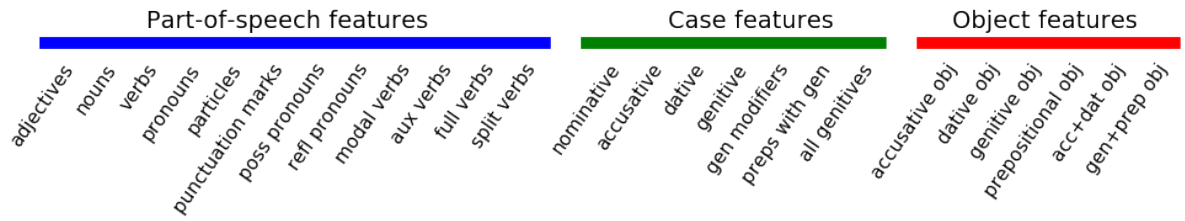


Figure 1: Part-of-speech, case and object features.

3.1.1 Part-of-speech features

The features in this category consist of the frequency of part-of-speech types: the 5 main classes *nouns*, *verbs*, *adjectives*, *pronouns* and *particles*, and more specific part-of-speech tags, for example modal verbs and auxiliary verbs (see Figure 1).

3.1.2 Case features

These features count the frequency of the four linguistic cases in German, or more precisely how many noun phrases can be found for each case. I added two specific features concerning the genitive case: the use of genitive modifiers and prepositions with genitives, as they are learned at different levels (see Figure 1).

3.1.3 Object features

The features in this category cover the use of different types of objects: accusative (direct) object, dative (indirect) object, genitive object and prepositional object, as well as two combinations (see Figure 1).

3.1.4 Tempus features

The features in this category reflect the use of the different tenses: present, perfect, past simple (*Präteritum*), past perfect (*Plusquamperfekt*), and future tenses (*Futur I* and *Futur II*). Additionally, I defined a feature for counting the use of past simple (*Präteritum*) forms of the auxiliary verbs *sein* and *haben*, as those forms are usually learned before properly studying the past simple (see Figure 2).

3.1.5 Mode features

The mode features show the use of different modes in a text, such as subjunctive mode and imperative mode. Since indicative mode is the most frequent and first-learned mode, I did not implement it as a feature. As for the subjunctive mode, I consider *Konjunktiv I* and *Konjunktiv II* forms, as well as the use of the auxiliary verbs *sein* and

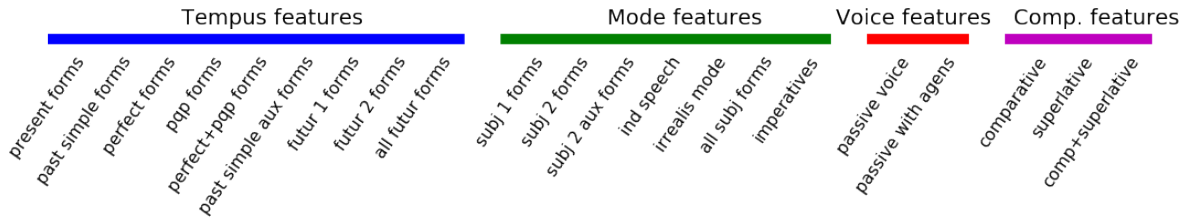


Figure 2: Tempus, mode, voice and adjective comparation features.

haben in *Konjunktiv II*. Derived from the subjunctive forms, I created two more specific features representing typical constructions of subjunctive mode: indirect speech and irrealis mood. Finally, the use of the imperative mood is also a feature of this category (see Figure 2).

3.1.6 Voice features

The passive voice yields two features: the frequency of passive voice in general and passive constructions that contain an agens, or logical subject (see Figure 2).

3.1.7 Adjective comparation features

The two features in this category represent the use of the comparative and superlative forms of adjectives (see Figure 2).

3.1.8 Conjunction and clause features

These features represent the use of conjunctions and clauses, for example the frequency of coordinating conjunctions or relative clauses. Further features reflect different kinds of subjunctive clauses, based on manually defined groups of subordinating conjunctions. The different types of subordinating clauses are causal, concessive, conditional, consecutive, final, interrogative, modal, temporal, and other. This feature category also contains participial clauses, in both present and past tense (see Figure 3).

3.1.9 Miscellaneous features

The first two features in this category consist of two constructions: ***brauchen*** `+ zu + infinitive`, and ***sich lassen*** `+ infinitive` (see Examples 1-2). Further features express the use of negations, questions, and the impersonal pronoun *es* with placeholder function (see Example 3). Another category shows how often the predicate and subject are inverted in a sentence, for example in questions or subjunctive clauses. Finally, there is a feature that represents the vocabulary density by counting the diversity of non-stopwords in the text. This feature is not normalized by dividing with the number of sentences (see Figure 3).

Figure 3: Conjunction and clause features, miscellaneous, entity and baseline features.

(1) Du brauchst nicht alle Aufgaben zu lösen.
You don't have to solve all the exercises.

(2) Ich lasse mich von meinem Bruder abholen.
I'll have my brother pick me up.

(3) Es wartet jemand auf Sie.
There is someone waiting for you.

3.1.10 Entity features

The entity features represent the number of entities per text and the average frequency of the entities. Since the individual entity frequencies are already normalized by the number of sentences, the average is not divided again. Coreferent discourse entities are considered as one entity. Coreference information is gained in the *CorZu* preprocessing step (see Figure 3).

3.1.11 Features for baselines

The two features used in the baseline system do not contain any syntactic information. The first feature simply represents the number of words per sentence. The second feature corresponds to the *Läsbarhetsindex* ('readability index'), also known as LIX score[13]. The LIX score computes the sum of the average sentence length and the ratio of long words (i.e., words with more than six letters) [Björnsson, 1968]. Naturally, the values of these features do not have to be normalized as they already contain normalization step (see Figure 3).

3.2 Combination of sparse features

Some of the described linguistic patterns are relatively rare. In order to reduce this feature sparsity, I combined several features into one. I assigned some of the features

[13] `http://www.psychometrica.de/lix.html`

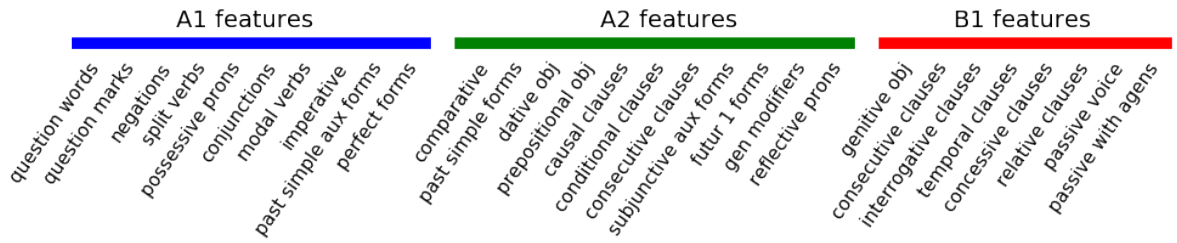


Figure 4: Combined features for levels A1, A2 and B1.

to one of three language level categories (A1-B1), according to the level at which the pattern or form in question is learned. 10 of the above features are associated with level A1, 12 with level A2, and 8 features are assigned to level B1 (see Figure 4). I did not a find sufficient number of exemplary features for level B2, so I refrained from creating a feature combination for that level.

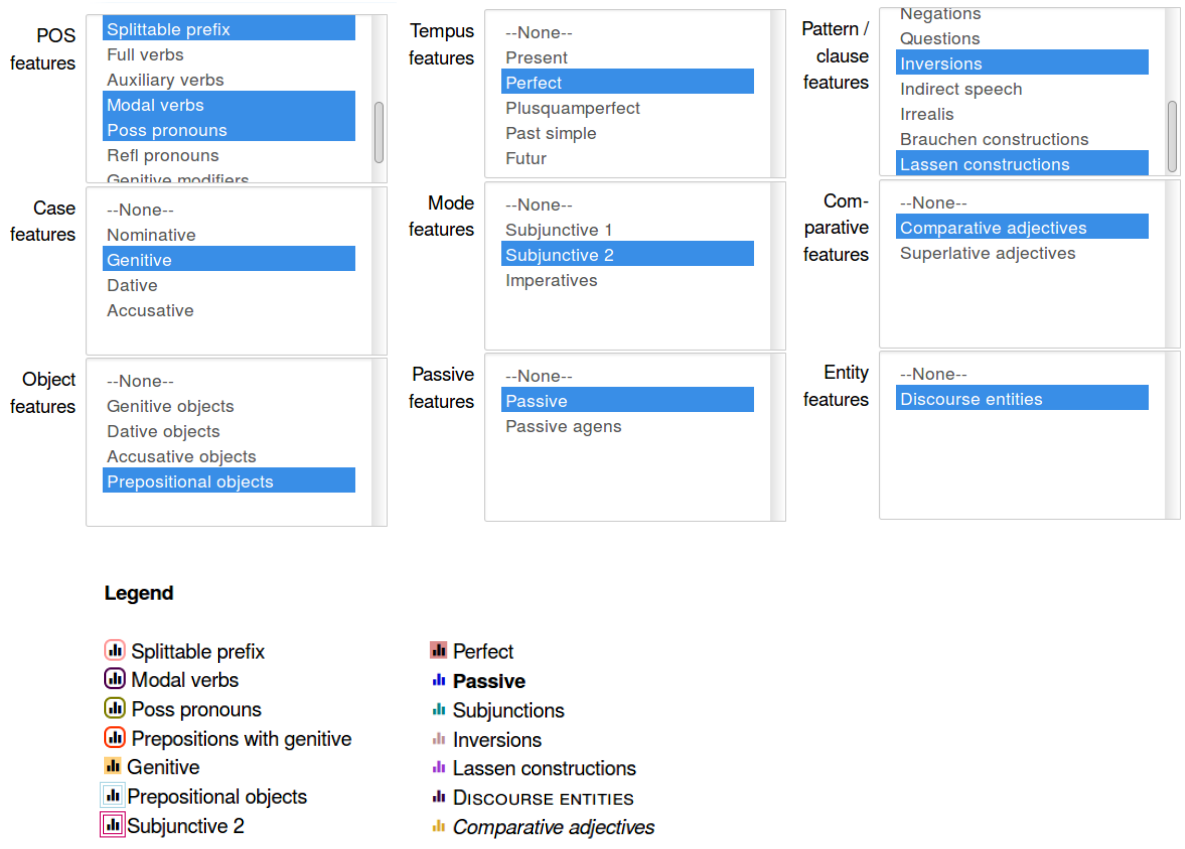
For each of the three language levels, the information of the associated features is combined, resulting in two features each: one sums the values of all features in the category and the other shows how many features have non-zero values, indicating if the pattern or form at all occurs in the text. In Chapter 4.2 a more general way of combining sparse features is introduced.

3.3 Visualization

For visualization of the features, I assigned HTML tags to the tokens in the text, which color them according to the pattern or phenomenon they represent. The system creates an HTML file for every input document. However, it is generally not useful to highlight all features at once since this yields a very colorful and confusing representation of the text. Instead, a user interface should provide the possibility of selecting a small set of desired features.

I employed Jupyter Notebook to implement a simple exemplary user interface that colors and marks the selected features according to their category. With this tool, any combination of linguistic elements in German texts can be visualized. In future work, this interface could be adapted to create a more advanced interactive tool that assists writing and evaluating texts in German, or controlled natural languages.

Figure 5 shows the interface and the colored output for a B2 level text. Note that mistakes made by the parser can occasionally lead to incorrect highlighting.



Wenn Freunde oder Bekannte um einen Gefallen bitten, sagt man oft „ Ja“, ohne lange nachzudenken. Denn man möchte den anderen nicht enttäuschen oder gar im Stich lassen, obwohl man der Bitte vielleicht gar nicht nachkommen möchte. „ Hilfst du mir beim Umzug?“ „ Was würdest du als Rechtsanwältin in diesem Fall machen?“ Oder, ganz klassisch: „ Du bist doch Arzt, kannst du dir nicht eben meine Schulter ansehen, mir tut es hier immer so weh...“ Aber manchmal ist es für die Freundschaft *besser*, wenn man einer solchen Bitte nicht nachkommt. Die Bitte um einen Gefallen ist besonders im Zusammenhang mit Beruflichem problematisch. So dürfte es einer Juristin zum Beispiel nicht unbedingt leichtfallen, fachlich fundierte Ratschläge zu geben, wenn man sie während eines Abendessens befragt. **Wird** der Freund oder Bekannte jedoch offiziell **beraten**, ist es schwierig, die Höhe des Honorars festzulegen. Falsch verstandene Tipps und anschließende Unstimmigkeiten bei der Bezahlung haben schon so mancher Freundschaft geschadet. Regelrecht gefährlich kann es unter Umständen werden, wenn ein Arzt am Küchentisch eine vage Diagnose stellt– und der Freund sich dann gar nicht mehr richtig untersuchen lässt oder sich im umgekehrten Fall ein Wochenende lang unnötige Sorgen macht. Daher ist es auf jeden Fall *besser*, wenn man Berufliches und Privates strikt auseinanderhält– und dann *lieber* eine kompetente Kollegin empfiehlt. Auch bei anderen Freundschaftsdiensten sollte man gut überlegen, ob man sie wirklich leisten möchte. Wenn man sich selbst nicht im Klaren ist, bleibt oft ein ungutes Gefühl zurück. Vor allem wenn sich jemand lange nicht gemeldet hat und dann nur anruft, weil er etwas von einem will, fühlt man sich unter Umständen ausgenutzt. In solchen Situationen sollte man von Fall zu Fall entscheiden, wie man sich verhält. Bei den Überlegungen, ob man den Freundschaftsdienst leisten will, sollte man immer berücksichtigen, wer um Hilfe bittet, was man für ihn tun soll und ob die Beziehung auf Gegenseitigkeit beruht. Damit eine Ablehnung nicht als Kränkung **empfunden wird**, sollte man sie als Ich-Botschaft formulieren und möglichst offen über die Situation sprechen, etwa: „ Weißt du, ich habe gerade so viel zu tun, ich kann dir leider nicht beim Umzug helfen.“ Was man auf keinen Fall tun sollte: Den anderen immer wieder hinhalten und vertrösten, wenn man eigentlich weiß, dass man der Bitte nicht nachkommen will. Das führt schnell zu Missverständnissen. Ein klares und begründetes „ Nein“ hingegen wirkt sich langfristig positiv auf die Beziehung aus, da Ehrlichkeit ja letztlich ein großer Freundschaftsbeweis ist.

Figure 5: Feature visualization example for B2 level text (*goethe_14.txt*).

4 Experiments

4.1 Classifier

In this chapter, I will describe how I built and optimized the language level classifier. The results are summarized in Tables 2 and 3 and discussed in Chapter 5. I employed the python machine learning library *scikit-learn* for implementation of a linear support vector classifier (SVC) [Pedregosa et al., 2011]. I chose a linear model based on the assumption that the levels A1-B2 are linearly separable since they represent increasing levels of language skills. Indeed, experiments show that a non-linear SVC with a radial basis function (RBF) kernel does not outperform a linear model (see Table 2). In such cases, the use of a simple linear SVC is highly advantageous as it takes considerably less training time and allows inspection of the coefficient of each feature, revealing which features contribute most to the prediction.

In the first experiments, all 80 features described in Chapter 3.1 are used. The features are standardized by centering the mean and scaling to unit variance. To identify the optimal values for the hyper-parameters C (penalty parameter C of the error term) and tol (tolerance for stopping criteria), I performed a hyper-parameter screening[14]. Since the dataset is relatively small I could not afford to define part of it as a development set, as this would have reduced the training set too drastically. Therefore, I directly used the training and test set for hyper-parameter screening, which entails some risk of overfitting, although linear SVCs are less susceptible to this problem. Indeed, hyper-parameter screening hardly changes the results compared to the default settings (see Table 2). Nevertheless, in future work the hyper-parameter screening should be executed on a development set that is not part of the dataset on which the classifier is ultimately trained and tested.

The classifier is trained on 90% of the data set and tested with the remaining 10%. In both training and test set the 4 categories are equally distributed. Precision, recall and F1 scores are reported as the weighted average over the 4 categories. The short computing time allows 50-fold cross-validation, which guarantees stable results. The top part of Table 2 shows the results for the baseline classifier and the classifier that employs all features, both with and without hyper-parameter screening. The baseline receives only the number of words per sentences and the LIX score as features.

4.2 Feature Agglomeration

Working with large number of features given a relatively small number of samples reduces the performance of classification algorithms, a problem commonly knowns as the *curse of dimensionality*. One way of solving this problem is feature agglomeration by hierarchical clustering. While agglomerative clustering is normally used to build clusters of similar samples, feature agglomeration instead uses it to merge together similar features. This approach can be implemented by clustering in the feature direction, in other words clustering the transposed data.

[14] $C = 0.6$ and $tol = 0.1$

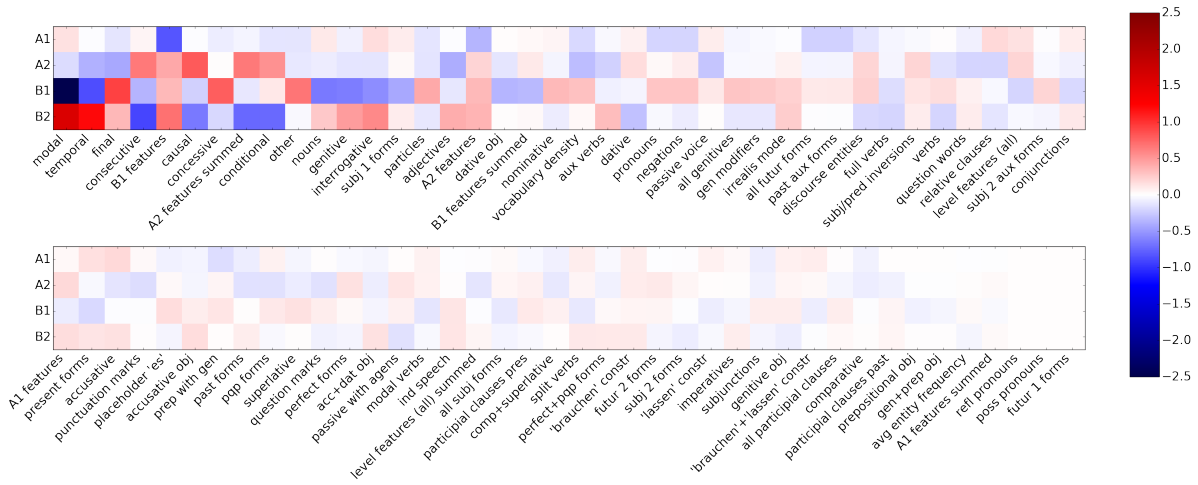


Figure 6: Features sorted by coefficient strength.

4.2.1 Agglomerating sparse features

I examined whether feature agglomeration of sparse features can improve classifier performance. I considered all features that have a 0 value in more than one third of the samples as sparse. This method classifies 43 features – more than half – as sparse. In the feature agglomeration process, those features are merged into 5 clusters. The original 43 features are discarded and the 5 new clustered features are included instead. Combined with the 37 non-sparse features, this yields a total of 42 features. This method counter-acts sparsity, reduces the number of features and mitigates the impact of noise. The feature agglomeration step led to an improvement in the classifier's performance (see Table 2).

4.2.2 Agglomerating less relevant features

Linear support vector classifiers allow inspection of the feature coefficients. They show how much a feature contributes to the prediction of each class. Figure 6 shows the complete set of features (without any agglomeration) sorted according to the strength of their coefficients: red indicates that a feature is considered a strong indicator for the class; blue means the feature works as a counter-indicator; white indicates that the feature is not informative for this class. Modal clauses, for instance, are considered strong indicators for B2 level texts, while they are unlikely to appear in B1 level texts. This could indicate that modal clauses are only learned at level B2. *Reflexive* and *possessive pronouns* and *futur 1 forms*, on the other hand, are not informative for any class at all. Note that the feature coefficients are determined based on the training set and can therefore be used for classifier optimization without any risk of distorting the assessment of classifier performance using the test set.

Only roughly the first 15 features show strong coefficients, suggesting that the other features contribute little to the prediction. An explanation for this observation could be feature sparsity: features that have the value 0 in most of the samples are less likely to be relevant when predicting a class. 9 of the 15 most relevant features are subjunctive clause features and 2 are combined level features. This supports the sparsity assumption as those features are naturally less likely to be sparse. However, sparsity does not necessarily make a feature irrelevant. The two sparse features *subjunctive 1 forms* and *genitives*, for example, are in the top 15 relevant features (see also Chapter 5.1). The remaining 2 features of the top 15 are two basic part-of-speech features: nouns and particles.

Using this information about feature relevance, I conducted another set of experiments to improve performance of the classifier. In the first, I only included the 15 most relevant features and discarded the others. In the second experiment, I applied feature agglomeration to merge the 65 less relevant features into 5 feature clusters and added them to the 15 relevant features. Both experiments showed similarly high results, clearly outperforming the default classifier with 80 features. Finally, I conducted experiments using only the less relevant features, both with and without feature agglomeration to demonstrate the impact of this method (see Table 2).

4.3 Datasets with 5 sentences

Since one of the short-comings of the project is the small dataset, I chunked the samples into 5 sentence fragments and treated them as independent samples to allow a preliminary investigation of the performance of my system on a larger dataset. This approach more than doubles the sample size from 264 to 614. Since the four categories are no longer well-balanced (184, 158, 121, and 151 samples), I decided to only take 121 of each class, resulting in a balanced set of 484 samples. However, the chunking of the original samples yields not only an increase in the number of samples but comes with two other effects. On the one hand, the samples are no longer independent, which could artificially improve classifier performance. On the other hand, one would expect a decrease in performance due to the reduced document length.

To assess this independently, I conducted further experiments with samples of sentence length 5, in which I extracted only a single chunk from each original document to guarantee independent samples. I created two different datasets in order to assess whether the beginning or middle of a text contains more language level relevant information. There are 19 samples with only 3 or 4 sentences which were included in the dataset with the first 5 sentences but discarded from that with the middle 5 sentences. All the experiments were conducted with the strongest classifier version, that is the version with feature agglomeration of the less relevant features. The results are shown in Table 3.

5 Discussion

Tables 2 and 3 show the results for all experiments detailed in Chapter 4. All approaches based on my syntactical features outperform the baseline. However, the baseline is still

	# Features	Accuracy ± SD	Precision	Recall	F1
Baseline	2	0.508 ± 0.092	0.490	0.508	0.453
Non-linear SVC (RBF)	80	0.856 ± 0.068	0.873	0.856	0.855
Linear SVC (not para. screened)	80	0.887 ± 0.060	0.899	0.887	0.886
Linear SVC	80	0.891 ± 0.058	0.902	0.891	0.890
Feat. aggl. (sparse f.)	42	0.911 ± 0.057	0.919	0.911	0.911
Rel. features only	15	0.967 ± 0.035	0.972	0.967	0.966
Feat. aggl. (less rel. f.)	20	0.962 ± 0.034	0.967	0.962	0.962
Less rel. features only	65	0.753 ± 0.079	0.767	0.753	0.747
Less rel. features feat. aggl.	5	0.846 ± 0.063	0.865	0.846	0.844

Table 2: Language level classification for baseline, default and feature agglomerated versions.

substantially better than simple guessing, which would yield only 25% accuracy, so the strength of the two baseline features should not be underestimated. The LIX score in particular is a measure for the readability of texts so it is not surprising that it is informative for predicting language levels, even if it is clearly not sufficient on its own.

The RBF-based non-linear SVC does not outperform the linear model. This finding is relevant when arguing in favor of the linear SVC approach, which is faster and yields information about the coefficients. The linear SVC including all 80 features is my default system.

Experiments showed that hyper-parameter screening does not change the results, suggesting that the classifier is highly robust. Nevertheless, in future work a large dataset that allows splitting off a development set would be desirable, also for systematic testing of other parameters, such as the cluster size in agglomerative clustering. Furthermore, a larger dataset is needed to statistically evaluate and compare the results of these experiments. For example, with a test set of only 27 samples in 4 classes, significance tests for a difference in performance between two systems (e.g. a binomial test on accuracy) may not be reliable.

5.1 Feature Agglomeration

Feature agglomeration on sparse features slightly improves the classifier's performance. The feature space is reduced from 80 to 42 features. Since many of the 80 features are sparse and thus do not contribute much to the class prediction on their own, merging them is a good method to reduce the number of features and handle feature sparsity.

An even greater improvement is shown in the experiments using information about the 15 most relevant features. In both experiments, the feature space is reduced to a quarter or less of the original size (15 and 20 features, respectively). Interestingly, the experiment with solely the 15 most relevant features yields results equally high as the

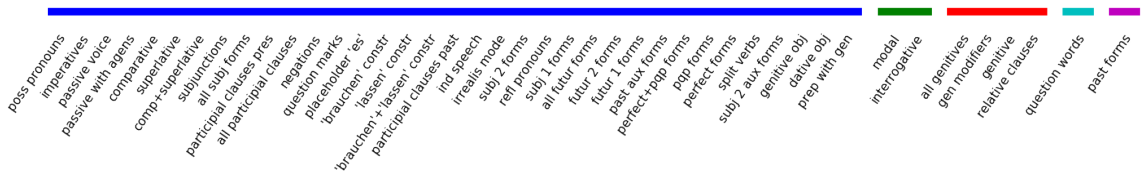


Figure 7: Feature clusters for sparse features.

experiment that adds the 5 features resulting from the feature agglomeration step of the less relevant features, implying that the 15 most relevant features already contain all information encoded in the others. However, this does not mean that the less relevant features are unsuitable for predicting language levels. The 65 less relevant features alone clearly outperform the baseline. Interestingly, the performance increases considerably when the same 65 features are merged into 5 features with feature agglomeration. Although consisting of the same input features, the classifier with agglomerated features outperforms its counterpart by almost 10% accuracy. This proves that feature agglomeration is a valuable preprocessing step, especially when dealing with a large number of features with high sparsity.

Note that the 15 features determined to be most relevant here may not all be relevant features in other classification tasks. However, the strategy of identifying relevant features based on SVC coefficients as well as the feature agglomeration approach for clustering less relevant features are generally applicable. Thus, this approach can be used in future work on other classification tasks to determine the most relevant features, reduce feature sparsity and noise, and further adjust the feature set.

5.1.1 Feature clusters

Since the feature agglomeration clusters the sparse or less relevant features into 5 clusters based on similarity, it is meaningful to analyze these clusters. Figures 7 and 8 show the 5 clusters for the experiments with sparse and less relevant features. The blue cluster in the sparse features experiment (Figure 7) includes 35 features; note that only one of these features (*subjunction 1 forms*) appears in the 15 most relevant features. The green cluster consists only of two subordinate clause features – modal and interrogative – which we already identified as relevant in Chapter 4.2.2. They occupy positions 1 and 13 in the relevance plot (see Figure 6). Modal and interrogative clauses, however, do not appear often so they were classified as sparse features. This proves that sparsity and relevance do not necessarily correlate; features can be strong indicators for a class despite, or perhaps because of, their sparsity. Modal clauses may be rare but when they do appear it is very likely a B2 level text; a similar but less pronounced effect can be observed for interrogative clauses. The green cluster shows the strongest coefficient in this experimental setting. Notably, the red cluster contains three features concerning

Figure 8: Feature clusters for less relevant features.

genitive case although the clustering algorithm does not know that they are indeed related. This may mean that the genitive features are not diverse enough, perhaps even redundant. The general *genitive* feature – which has also proven to be a relevant feature – may be sufficient. The remaining clusters with only *questions words* and *past forms* respectively contain less relevant features. Those singleton clusters may result from the request for 5 clusters. Reducing the number of clusters could improve results and reduce more noise. Future work should attempt to optimize this hyper-parameter.

Figure 8 gives the feature agglomeration results for the experiments with less relevant features. The blue cluster contains 19 features, mostly from the categories *part-of-speech*, *case*, *object* and *entitiy features*. The green cluster includes 41 features, representing mostly the categories *tempus*, *mode*, *(passive) voice*, *adjective comparation* and *miscellaneous features*. The magenta cluster contains the combined features for A1 and A2, as well the combined features for A1-B1 altogether. The cluster merged those features without having the information that they are actually closely related. While *A2 features* just missed the top 15 relevant features (position 16), the other features are located in the second third of the relevance scale. *B1 features* is considered relevant as well, which proves that the combination of features by level is an efficient method, although it may not work equally well for every level. Note that this cluster merges with the blue one when the cluster number is reduced to 4. The two singleton clusters with *B1 features summed* and *vocabulary density* also merge with the blue cluster when the cluster size is further reduced to 3 or 2, respectively. It is not surprising that *vocabulary size* constitutes a relatively strong singleton cluster since it is a very exclusive feature with little similarity to others. Further inspection of the clusters may help defining more suitable and less redundant features in the future.

5.2 Datasets with 5 sentences

Only taking 5 sentences per samples works surprisingly well (see Table 3). Apparently, a chunk of 5 sentences captures enough information to allow language level classification and outperform the baseline by far. The effect is even more visible when considering the middle 5 sentences instead of the first 5, although the sample size is reduced by 7%. This may suggest that the middle of a document contains more information relevant to language level classification than the beginning.

	# Samples	Accuracy ± SD	Precision	Recall	F1
Feat. aggl. (less rel. f.)	264	0.962 ± 0.034	0.967	0.962	0.962
Only first 5 sents	264	0.904 ± 0.058	0.914	0.904	0.903
Only middle 5 sents	245	0.934 ± 0.042	0.942	0.934	0.932
Texts split into 5 sents	484	0.973 ± 0.022	0.975	0.973	0.973

Table 3: Language level classification results for 5 sentences datasets.

Splitting the texts into chunks of 5 sentences considerably increases the sample size. The performance is similarly high as in the experiments with the complete samples but these results should be interpreted with care. Although it seems that the increased sample size makes up for the information lost by the 5 sentences limitation, the increased homogeneity of the dataset may lead to an overestimation of classifier performance. For future experiments and further development, a larger dataset of independent samples is therefore indispensable in order to overcome the potential problems introduced by a small sample size.

6 Conclusion

In this work, I have presented my system for analyzing German texts in order to determine the language level. I have introduced a number of linguistic features that are valuable for the analysis of language levels. I have developed a system that counts and marks them in order to assist manual evaluation. For visualization, I have implemented a simple interactive tool that allows selection of features, which are then colored and marked in the text.

Furthermore, I have taken these features and trained a linear support vector classifier that is able to differentiate between the levels A1 to B2 with an accuracy of 89%. I have shown how the inspection of feature sparsity and relevance can reveal valuable information about how to improve the classifier. I have employed feature agglomeration to reduce the feature space, which improved the performance to 96% accuracy. I have examined the feature clusters that resulted from feature agglomeration. Further, I have demonstrated that fragments of 5 sentences are sufficient for language level classification. I have pointed out the short-comings of my work, in particular the small dataset, and recommended solutions for overcoming them. Furthermore, I have discussed possible applications for my system or adaptations of it and suggested ideas to further develop and improve this work in the future.

My work shows that a system exclusively based on linguistic features extracted from a dependency parser can efficiently and accurately predict the language level of a given text. The system gives automatic feedback on the syntactic complexity of a text and therefore assists language learners as well as teachers.

References

Becker, L., Basu, S., and Vanderwende, L. (2012). Mind the Gap: Learning to Choose Gaps for Question Generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 742–751. Association for Computational Linguistics.

Björnsson, C. H. (1968). Läsbarhet, Stockholm: Liber.

Bock, B. M. (2014). Leichte Sprache: Abgrenzung, Beschreibung und Problemstellungen aus Sicht der Linguistik. *Sprache barrierefrei gestalten. Perspektiven aus der Angewandten Linguistik. Berlin: Frank & Timme.(= TransÜD. 69)*, pages 17–52.

Burstein, J., Chodorow, M., and Leacock, C. (2003). CriterionSM Online Essay Evaluation: An Application for Automated Evaluation of Student Essays. In *IAAI*, pages 3–10.

Dengler, S. and Sieber, T. (2017). *Netzwerk Grammatik A1-B1; Deutsch als Fremdsprache. Übungsbuch*. Klett Sprachen GmbH.

Dikli, S. (2006). Automated Essay Scoring. *Turkish Online Journal of Distance Education*, 7(1).

DuBay, W. H. (2004). The Principles of Readability. *Online Submission*.

Flesch, R. (1948). A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Griesshaber, W. (2005). Sprachstandsdiagnose im kindlichen Zweitspracherwerb: Funktional-pragmatische Fundierung der Profilanalyse. *Veröffentlichungsort unbekannt.*, pages 1–58.

Heilman, M. and Smith, N. A. (2010). Good Question! Statistical Ranking for Question Generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.

Klaper, D., Ebling, S., and Volk, M. (2013). Building a German/Simple German Parallel Corpus for Automatic Text Simplification. In *Proc. of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 11–19.

Klenner, M. and Tuggener, D. (2011). An Incremental Entity-Mention Model for Coreference Resolution with Restrictive Antecedent Accessibility. In *RANLP*, pages 178–185.

Kuhn, T. (2014). A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, 40(1):121–170.

Leacock, C. and Chodorow, M. (2003). C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities*, 37(4):389–405.

Maaß, C. and Bredel, U. (2016). *Leichte Sprache: Theoretische Grundlagen, Orientierung für die Praxis*. Bibliographisches Institut GmbH.

Nielsen, R. D., Ward, W., and Martin, J. H. (2009). Recognizing Entailment in Intelligent Tutoring Systems. *Natural Language Engineering*, 15(4):479–501.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sennrich, R., Schneider, G., Volk, M., and Warin, M. (2009). A New Hybrid Dependency Parser for German. *Proceedings of the German Society for Computational Linguistics and Language Technology*, 115:124.

Shermis, M. D. and Burstein, J. C. (2003). *Automated Essay Scoring: A Cross-Disciplinary Perspective*. Routledge.

Štajner, S., Popovic, M., Saggion, H., Specia, L., and Fishel, M. (2016). Shared Task on Quality Assessment for Text Simplification. *Training*, 218(95):192.

Suter, J., Ebling, S., and Volk, M. (2016). Rule-based Automatic Text Simplification for German. *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, pages 279–287.

Witzlinger, H. (2017). Deutsch; Aber Hallo! Grammatikübungen. `http://www.deutschkurse-passau.de/JM/index.php?option=com_content&view=article&id=34&Itemid=165`. Last checked on 28.11.2017.