# Semantic Text Segment Classification of Structured Technical Content

Julian Höllig[1](✉), Philipp Dufter[2], Michaela Geierhos[1], Wolfgang Ziegler[3], and Hinrich Schütze[2]

[1] Research Institute CODE, Bundeswehr University Munich, Neubiberg, Germany
{julian.hoellig,michaela.geierhos}@unibw.de
[2] Center for Language and Information Processing, LMU Munich, Munich, Germany
philipp@cis.lmu.de
[3] Information Management and Media, Karlsruhe University of Applied Sciences, Karlsruhe, Germany
wolfgang.ziegler@hs-karlsruhe.de

**Abstract.** Semantic tagging in technical documentation is an important but error-prone process, with the objective to produce highly structured content for automated processing and standardized information delivery. Benefits thereof are consistent and didactically optimized documents, supported by professional and automatic styling for multiple target media. Using machine learning to automate the validation of the tagging process is a novel approach, for which a new, high-quality dataset is provided in ready-to-use training, validation and test sets. In a series of experiments, we classified ten different semantic text segment types using both traditional and deep learning models. The experiments show partial success, with a high accuracy but relatively low macro-average performance. This can be attributed to a mix of a strong class imbalance, and high semantic and linguistic similarity among certain text types. By creating a set of context features, the model performances increased significantly. Although the data was collected to serve a specific use case, further valuable research can be performed in the areas of document engineering, class imbalance reduction, and semantic text classification.

**Keywords:** semantic text classification · context features · technical documentation.

## 1 Introduction

The area of technical documentation is highly relevant in the industry, due to the legal need for technical information, such as user manuals, alongside commercial products [4]. There are established standards to ensure efficiency and quality in the document creation process. One important standard is the uniform assignment of XML tags to text segments. Some of these segments, such as notes, commands or warnings, contain semantics, while others, like continuous text or generic lists, are non-semantic. Semantic text segments are indicated by the XML name tag. Depending on the communicative goal of the segment type,

the respective tags are connected to rules, which ensure consistent structure and layout in the documentation [4]. Besides supporting readability of the contents, this provides the prerequisites for intelligent information processing.

When multiple authors, eventually located at multiple sites, write documents for the same customer, or even work together on a single document, divergent tag assignments are likely to happen. Some text segments are semantically very close, which might as well lead to different tagging. Our research focuses on the automated validation of the tagging process through the support of machine learning, and thus on the improvement and standardization of highly structured content, including all the associated benefits.

There is much prior research on text classification, especially on applications for social media [5,15,2], and online product customer reviews [7,11,6]. As opposed to this type of content, technical content is highly structured, emotionally unbiased, and usually follows writing guidelines. Consequently, important features for the classification remain in the communicative style, and in contextual patterns. Therefore, we extracted a set of context features specific to structured content, and tailored to the standards of the data source.

Our study makes several contributions: first, we developed a new concept for validating automated tagging, which enables intelligent and automated information processing; second, we created a comprehensive, high-quality dataset, as a basis for further research on the use case, or similar scenarios; third, we designed context features to increase model performance and to save resources.

## 2 Related Work

Text classification is a common area in computational linguistics and has been applied to diverse domains such as health [2], law [8], finance [20], and social media [5,15]. The text input size reaches from document, chapter, and paragraph, to even sentence level. The few works done at the interface between machine learning and technical documentation deal with classification of relatively large, self-contained units of content, for example on chapter [12,13] or document level [9]. Writing in all conscience, this work is the first dealing with paragraph-level text classification in the technical documentation domain.

Oevermann and Ziegler [13] conducted a comprehensive study, where they applied traditional machine learning models to categorize product component-related text blocks in technical documents. They also used real-world data, which was manually tagged by professionals such as technical writers, or content experts. While they used data from the engineering sector, the data for our work is software-related. They applied the vector space model as baseline, and tf-idf weighting for single words and word groups, where word n-grams of two and three achieved the best performances. The classification was done by finding the highest cosine similarity between a document vector and the class vectors.

Since the invention of BERT in 2018 [1], the model has been frequently applied to various text classification tasks and compared to traditional machine learning methods, which mostly used tf-idf feature extraction. The superior-

ity of BERT was found in most cases. González-Carvajal and Garrido-Merchán [6] conducted several binary- and multi-classification tasks on movie and hotel reviews, where BERT achieved better performance than traditional models, including a support vector machine (SVM). In our work, we use similar-sized text data, and we apply the same models, but to a fairly unexplored domain. Lund [9] contradicts the results found by González-Carvajal and Garrido-Merchán [6] by achieving parity of a tf-idf model and BERT. Lund applied BERT to technical documents following the product life cycle such as installation, operation, maintenance, troubleshooting, and disposal. In our work, we also match BERT's performance with a tf-idf model, but only by using additional context features.

Di Iorio et al. [3] followed the goal of standardized document structures to achieve consistent layouts. But instead of applying machine learning, they approached the task with an algorithm based on a pattern rules concept. This concept was developed by abstracting XML patterns from large amounts of documents. Hereby, the authors identified common structural and content-related characteristics in and between typical XML elements that applied to all documents. Unlike in our work, the semantic value of the contents was irrelevant. Examples for patterns were blocks, containers, or fields. While the authors considered structural patterns as the basis for styling decisions, we are considering the semantics as more important. For instance, our documents contain tables with entries of definitions, and while we would capture the actual definitions, the authors from this work would capture the whole table. Due to the strong abstraction, this approach is not as fine-granular as ours, but generalization can be achieved more easily.

## 3 Data

### 3.1 Data Collection

**Data Source.** The dataset for this work was scraped from the SAP Help Portal[4], an open-source online documentation platform containing a high number of user manuals for different SAP products. The documentation is created and maintained in a content management system, and structured according to DITA.[5] DITA is a popular XML standard, which is frequently used in the technical documentation industry.

**Segment Type Definition.** We defined ten text segment types by manually examining documents in the SAP Help Portal across different products. Hereby, we used the underlying CSS classes in the HTML code to identify the different segment types. The following semantic text segment types are contained in the dataset: Command, Definition, Example, Note, Recommendation, Reminder, Restriction, Tip, Warning, Shortdescription.

[4] https://help.sap.com/viewer/index
[5] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita

**Data Quality.** Although in no official cooperation, SAP confirmed the defined segment types and gave insight in their content creation process. The contents in the SAP Help Portal are written by experts in the field of technical writing, who are supported by an editorial guide stating tagging standards and writing style recommendations. After data collection, we reviewed 100 random samples of each segment type to validate the data quality. Review criteria were the semantic correctness and data cleanness. Apart from a few outliers (text missing, incorrect tag assignments), the quality was good.

**Scraping Process.** For the data collection process, we built a web scraper using the Python framework *selenium*. Selenium offers user-like interaction with web content by taking control of the browser [17]. The segment types were identified in the underlying CSS class of the web page, and retrieved via XPath expressions. Example 1 illustrates this process.

*Example 1.* The example shows how to scrape segment type *Warning*.

```
driver.find_elements_by_xpath("//section/descendant::aside[@class =
    'note note caution']")
```

The object `driver` is a WebDriver element at the currently active browser window. The function `find_elements_by_xpath` finds the required elements via a XPath expression. In this example, the driver traverses the DOM tree and looks for all HTML elements descending from *section*, being tagged with *aside* and containing the class attribute *note note caution*.

### 3.2 Dataset Description

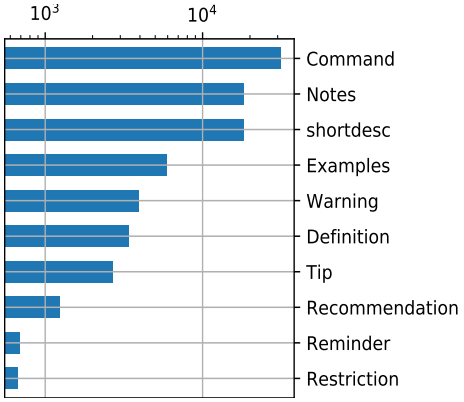
**Data Statistics.** The dataset was randomly split into training (70%), test (20%), and validation (10%) sets. Figure 1(a) shows the count of collected text segments per class (=type), and for each set, with a total count of **86,450**, after postprocessing. There are huge divergences between the class counts, which are visualized in Figure 1(b). This strong class imbalance caused issues in the following classification experiments.

**Data Access.** We provide the datasets on Github[6] in three json files (train.json, val.json, test.json) in postprocessed form. On request, we also provide all data before preprocessing in one big XML file, which contains all semantic and non-semantic segments collected. Along with the XML file, we provide a Python module *dataset.py* for data cleaning, sampling, context feature extraction, and transformation into *pandas* data frames. The data cleaning involves removal of inline HTML tags, HTML-generated newlines / tabs / spaces, 'None' values, and duplicates. The sampling involves oversampling on minority and undersampling on majority classes to enable the model to learn small classes [18].

[6] https://github.com/juhoUnibw/semSegClass

| Total count: **86,450** | Class counts | | |
|---|---|---|---|
| | Train | Val | Test |
| Commands | 20,081 | 2,231 | 9,180 |
| Notes | 11,604 | 1,289 | 5,336 |
| Shortdescriptions | 11,498 | 1,278 | 5,409 |
| Examples | 4,746 | 528 | 625 |
| Warnings | 3,123 | 347 | 459 |
| Definitions | 2,756 | 306 | 368 |
| Tips | 2,151 | 239 | 291 |
| Recommendations | 981 | 109 | 156 |
| Reminders | 558 | 62 | 69 |
| Restrictions | 529 | 59 | 82 |

(a) Class counts for training, validation and test sets




(b) Relative class distributions (logarithmic scale)

Fig. 1: Dataset: overview of class counts and distributions

## 4 Methods

### 4.1 Feature Extraction

In addition to the presented text segments, we developed nine context features, which were added to all models in different combinations. They can be categorized in topical, structural, environmental, and grammatical features. Table 1 states the categories, underlying features, and short explanations. All features were normalized for values between zero and one.

### 4.2 Models

**Model Selection.** We evaluated the use case through a series of experiments with a deep learning and a traditional model. Hereby, we applied the transformer model BERT (Bidirectional Encoder Representations from Transformers) from the *transformers* library by Huggingface [7], and a linear SVM (Support Vector Machine) from the *sklearn* library. As deep learning model, we chose BERT due to its state-of-the-art performances in many text classification tasks [16]. BERT also uses the so-called *self-attention* mechanism to capture long-range dependencies in text sequences [19], which we assumed to be helpful in finding complex writing style patterns. As traditional model we chose linear SVM because it shows the best results for text classification tasks between several traditional models [10]. Alternative deep learning and traditional models such as ALBERT, Gradient Boosting, Decision Tree, and Bagging performed worse than the presented models in first experiments, which is why they were deprecated.

[7] https://huggingface.co/transformers/model_doc/bert.html#tfbertforsequenceclassification

Table 1: Categorization and explanation of extracted context features

| **Category** | **Feature** | **Explanation** |
|---|---|---|
| Topical | TF | =Text Function. States the text function of the chapter where the segment was found (e.g. instructing, descriptive). Represented as binary feature over all text function categories (concept, task, reference, topic). |
| | chapTitle | =Chapter Title. Extracts a tf-idf representation of the chapter title where the segment was found. |
| Structural | ST | =Sibling Types. States the predecessor and successor segment types of the current segment. Represented as binary feature over all segment types. |
| | segPos | =Segment Position. States the position of the current segment within the chapter (0=start, 1=end). |
| | chapPos | =Chapter Position. States the position of the current segment within the document (0=start, 1=end). |
| Environmental | nSeg | =Number of Segments. States the number of segments within the chapter where the segment was found. |
| | CS | =Content Share. Measures how much of the chapter content is owned by the segment (chars segment / chars chapter). |
| | semDistr | =Semantic Distribution. Measures the semantic quantity and diversity within the chapter where the segment was found (number of semantic segments + number of unique semantic segments). |
| Grammatical | POS | =Part-of-speech. Extracts a tf-idf representation of the segment text after it was transformed into part-of-speech tags. |

**Model Design.** In order to use the pre-trained BERT model, a classification layer was built on top of the traditional BERT architecture. During training, the additional context features were appended to the output of the pooler layer of BERT, which is the sequence representation fed into the classification layer. The weight embedding matrix was adapted to the increased number of features. To access the pooler layer, the source code of the feed forward function was extracted from the transformers library and modified accordingly. For the SVM, the context features were appended as numpy array to the text representation.

**Hyperparameter Selection.** Table 2 shows the hyperparameter configurations we used with the models. BERT's configuration is recommended by Akshay Prakash [14]. For the linear SVM model, we validated different combinations to find the optimal one.

### 4.3 Sampling

An oversampling of factor 2 was applied to the minority classes (*Definition, Example, Warning, Tip, Recommendation, Reminder, Restriction*), and an un-

Table 2: Hyperparameters used with BERT (left) and SVM (right)

| **BERT** | |
|---|---|
| Batch size | 16 |
| Learning rate | 2e-5 |
| Epochs | 4 |
| Max. sequence length | 128 |
| Loss function | Cross-entropy |

| **Linear SVM** | |
|---|---|
| Stemming | Yes |
| Stop words | English |
| Tolerance | 0.0001 |
| Max. features | 20,000 |
| Loss function | Hinge-loss |

dersampling factor of 0.6 to the majority classes (*Command, Note, Shortdescription*). The samplings were only applied to the training and validation sets, the test set retains the real-world data distribution.

## 5 Experiments

### 5.1 Setup

In a series of experiments with eleven setups, we evaluated the classification of the ten presented semantic segment types through application of the presented models. In setup 1, bare text input was used for modeling. In setups 2-10 the impacts of the presented context features were evaluated. In setup 11, the best combinations of features for BERT and SVM were modeled. For the SVM, we additionally modeled bare context features, to compare their performance against bare text features.

The experiments were evaluated both quantitatively and qualitatively. In the quantitative evaluation, we present the overall performances of each experiment, measured in accuracy (1) and macro-average (5). The accuracy specifies the fraction of correctly classified text segments, the macro-average reflects the average success of all classes. We also specify the individual performances of each class in F1-score (4) for the best models. The exact definitions of the measures are as follows (where TP stands for True Positives, FP for False Positives, FN for False Negatives, N for the count of test samples, and C for the number of classes):

$$Accuracy = \frac{TP + TN}{N} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

$$Macro\text{-}average = \frac{\sum_{i=1}^{C} F1}{C} \tag{5}$$

In the qualitative evaluation, we reveal some challenging aspects of the classification task by analyzing the test samples that were hard to distinguish for the model.

### 5.2 Quantitative Evaluation

Table 3 shows the overall model performances of all experiments. The highest macro-average (60%) was achieved by BERT combined with the context features 'Text Function' and 'Siblings Type'. The highest accuracy (88%) was achieved by SVM combined with all context features. Overall, these models match in performance, which indicates that superior text embedding in deep learning models can be equalized by using context features in traditional models.

Table 3: Overall model performance comparison across all setups (values in %)

| | Accuracy | | Macro AVG | |
|---|---|---|---|---|
| | BERT | SVM | BERT | SVM |
| Setup 1: Text only | 83 | 77 | 54 | 45 |
| Setup 2: +TF | 84 | 79 | 59 | 48 |
| Setup 3: +chapTitle | 84 | 78 | 54 | 50 |
| Setup 4: +ST | 83 | 87 | 54 | 54 |
| Setup 5: +segPos | 83 | 79 | 56 | 46 |
| Setup 6: +chapPos | 83 | 77 | 56 | 46 |
| Setup 7: +nSeg | 83 | 77 | 54 | 46 |
| Setup 8: +CS | 84 | 77 | 55 | 45 |
| Setup 9: +semDistr | 83 | 77 | 54 | 47 |
| Setup 10: +POS | 84 | 78 | 54 | 46 |
| Setup 11: +Best combination | 84 | **88** | **60** | 56 |
| Only context features | - | 85 | - | 46 |

Table 4 shows the individual class performances of the two best models. The five best classes, which are the same for both models, are marked in bold. The other classes show low performances, due to high semantic (and linguistic) similarities, and the negative class imbalance influence. These challenges are further examined in the following subsection.

### 5.3 Qualitative Evaluation

Here, we examined individual test predictions to understand the challenges of the classification task. The basis of our analysis was 100 random test samples of each segment type, of which 50 samples were correctly, and 50 samples were incorrectly predicted. In the following, we focus on examples of conflicting segment type pairs, where the one type was mispredicted as the other type.

Table 4: Class performances (values in %) of the two best models: BERT and SVM from setup 11 (PREC=Precision, REC=Recall, F1=F1-score)

| | BERT+TF+ST | | | SVM+ALL | | | |
|---|---|---|---|---|---|---|---|
| | PREC | REC | F1 | PREC | REC | F1 | Count |
| **Command** | 95 | 94 | 95 | 94 | 96 | 95 | 9,180 |
| **Definition** | 87 | 69 | 77 | 96 | 98 | 97 | 368 |
| **Example** | 75 | 78 | 77 | 60 | 72 | 66 | 625 |
| **Note** | 77 | 76 | 77 | 81 | 75 | 78 | 5,336 |
| Recommendation | 46 | 45 | 45 | 35 | 33 | 34 | 156 |
| Reminder | 71 | 30 | 42 | 14 | 10 | 12 | 69 |
| Restriction | 41 | 28 | 33 | 42 | 22 | 29 | 82 |
| Tip | 30 | 39 | 34 | 30 | 26 | 28 | 291 |
| Warning | 34 | 44 | 38 | 23 | 25 | 24 | 459 |
| **Shortdescription** | 85 | 85 | 85 | 97 | 98 | 97 | 5,409 |
| Accuracy | **84** | | | **88** | | | **21,975** |
| Macro AVG | **60** | | | **56** | | | |

**Command vs. Note.** Example 2 shows the command-like syntax of type *Command*, but the wording legitimates the type *Note*, due to the phrase 'Make sure'. Without further context, it is difficult to say whether the true class *Command* or predicted class *Note* should actually be correct.

*Example 2.* "Make sure the SNC PSE is still the selected PSE."

**Shortdescription vs. Definition.** *Shortdescription* and *Definition* both contain descriptions of some kind, which is why they have similar linguistic patterns. The sample in Example 3 of type *Definition* could also be used in a *Shortdescription* segment, for example, to introduce a chapter.

*Example 3.* "The options that describe the operation of an object, which are viewable in the workspace when you open the object."

**Note vs. Warning.** Generally, the *Note* type is semantically very similar to the other note-like classes *Tip, Recommendation, Restriction, Reminder*, or *Warning*. The terms 'should' and 'have to' in Example 4 justify the falsely predicted *Warning* as well as the correct type *Note*.

*Example 4.* "The following checks and steps should be performed on all hosts of the affected sap HANA system. They have to be executed as the root user in the Linux shell."

**Restriction vs. Note.** Example 5 shows a sample of the type *Restriction*, which was predicted as *Note*. In this case, one could argue that the term 'available only' indicates a restriction. However, the models were trained with far more *Note* than

*Restriction* samples (11,604 : 529), so that a single linguistic difference like this seems to be not strong enough to influence the model.

*Example 5.* "The feature is available only on browsers (desktop / laptop)."

**Tip vs. Note.** A similar effect can be observed for *Tips*, which often contain the indicator 'you can' in order to animate the reader to act. Example 6 shows such a case, where the sample was predicted as *Note*. In this example, we can observe a mix of segment types, which would be legitimate, but hard to learn for the model. The last sentence for itself could easily belong to type *Command*.

*Example 6.* "If you have one instructor and you want to authorize that one instructor to teach many learning items, you can do that in the instructor's record. Go to people instructors authorized to teach."

**Reminder vs. Command.** A challenge of the type *Reminder* is that it can easily be formed by just repeating any statement made at some point in the documentation, while changing the semantics of that statement. The statement in Example 7 shows all linguistic patterns of a *Command* (verb at the start of sentence, imperative form), but the author might have tagged it legitimately as *Reminder* to prevent the reader from missing an action.

*Example 7.* "Copy and save the client secret as you won't be able to retrieve it later."

**Discussion.** Most of the challenging test samples belong to one of the sub-note types because they are hard to distinguish from the general *Note* type, both linguistically and semantically. This can be supported by the fact that 57.6% of the mispredicted note sub-type samples were falsely predicted as *Note*. The strong class imbalance in the data adds additional complexity to the classification of these types. In an experiment with equal class distribution, the performances of the sub-note type classes increased, but still remained below the other classes. Consequently, both the class imbalance and close class similarity affect their classification results negatively. Selective random oversampling, multiple SMOTE variants, class re-weighting, and a feature selection method did not improve the performance. The most effective solution to both problems is to merge all note type classes, taking into account the restraints it puts on the use case. For this scenario, the performance for the best model achieved 93%/89%.

## 6 Conclusion and Future Work

In this paper, we introduced the novel approach of validating document structures by means of machine learning in order to enable intelligent information processing and ensure consistent document layouts. We showed model evaluations with promising results, and revealed the remaining challenges. Moreover,

we provided a comprehensive dataset for further research in different areas such as document engineering, text classification, and the handling of imbalanced data, along with baseline results. During the experiments, we discovered the strong impact of context features on the performance of traditional models. Our SVM model, which was originally thought of as baseline model, matched BERT's performance through using context features. We could derive that structured semantic content yields useful underlying contextual patterns besides linguistic features. Thus, choosing traditional models with context features over deep learning models in such a scenario can achieve the same results with significantly less resources.

Our experimental results showed that the best deep learning model (BERT) and the best traditional model (SVM) achieve equal performances. They solve the classification task partially well, with a combined macro-average performance of 83.4% for the classes *Command, Definition, Example, Note,* and *Shortdescription*, and of 31.8% for the classes *Recommendation, Reminder, Restriction, Tip,* and *Warning*. Hereby, the SVM model achieves better performance on the first group of the classes (86.6%), while BERT achieves better performance on the second group of the classes (38.2%).

A big constraint of this work is the semantic and linguistic similarity between the note type elements. Combined with the class imbalance, the generic majority class *Note* is mostly predicted in unclear cases. Merging the sub-note types into a common note-type class, shows the potential of the application, and produces a ready-to-use model for the use case, although restricted to fewer classes.

A substantial advancement of our system would be the automated tagging of unstructured documents, for example, in the context of migration of large document collections to content management systems. Such an application would significantly lower the initial workload of content structure standardization in the industry and therefore, accelerate the process of intelligent content processing and delivery.

**Acknowledgments.** This work was supported by the Bavarian Research Institute for Digital Transformation and the European Research Council (#740516).

## References

1. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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). pp. 4171–4186. ACL, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423
2. Dhiman, A., Toshniwal, D.: An Enhanced Text Classification to Explore Health based Indian Government Policy Tweets. CoRR **abs/2007.06511** (2020)
3. Di Iorio, A., Peroni, S., Poggi, F., Vitali, F.: A first approach to the automatic recognition of structural patterns in XML documents. In: Concolato, C., Schmitz, P. (eds.) ACM Symposium on Document Engineering,

DocEng '12, Paris, France, September 4-7, 2012. pp. 85–94. ACM (2012). https://doi.org/10.1145/2361354.2361374
4. Drewer, P., Ziegler, W.: Technische Dokumentation: Übersetzungsgerechte Texterstellung und Content-Management, pp. 25–27. Vogel Business Media (2011)
5. Fei, G., Liu, B.: Social Media Text Classification under Negative Covariate Shift. In: Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., Marton, Y. (eds.) Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. pp. 2347–2356. ACL (2015). https://doi.org/10.18653/v1/d15-1282
6. González-Carvajal, S., Garrido-Merchán, E.C.: Comparing BERT against traditional machine learning text classification. CoRR **abs/2005.13012** (2020)
7. Gräbner, D., Zanker, M., Fliedl, G., Fuchs, M.: Classification of Customer Reviews based on Sentiment Analysis. In: Fuchs, M., Ricci, F., Cantoni, L. (eds.) ENTER 2012. pp. 460–470. Springer, Vienna (2012)
8. Lee, J.S., Hsiang, J.: Patent Classification by Fine-Tuning BERT Language Model. World Patent Information **61**, 101965 (2020). https://doi.org/10.1016/j.wpi.2020.101965
9. Lund, M.: Duplicate Detection and Text Classification on Simplified Technical English. Dissertation, Linköping University (2019), http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-158714
10. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK (2008)
11. Nicholls, C., Song, F.: Improving sentiment analysis with Part-of-Speech weighting. In: 2009 International Conference on Machine Learning and Cybernetics. vol. 3, pp. 1592–1597 (2009). https://doi.org/10.1109/ICMLC.2009.5212278
12. Oevermann, J.: Reconstructing Semantic Structures in Technical Documentation with Vector Space Classification. In: Martin, M., Cuquet, M., Folmer, E. (eds.) SEMANTiCS 2016, Leipzig, Germany, September 12-15, 2016. CEUR Workshop Proceedings, vol. 1695. CEUR-WS.org (2016)
13. Oevermann, J., Ziegler, W.: Automated Classification of Content Components in Technical Communication. Computational Intelligence **34**(1), 30–48 (2018)
14. Prakash, A.: Fine-Tuning BERT model using PyTorch (December 2019), https://medium.com/@prakashakshay90/f34148d58a37
15. Pratama, B.Y., Sarno, R.: Personality Classification Based on Twitter Text Using Naive Bayes, KNN and SVM. In: 2015 International Conference on Data and Software Engineering (ICoDSE). pp. 170–174 (2015). https://doi.org/10.1109/ICODSE.2015.7436992
16. Raj, B.S.: Understanding BERT: Is it a Game Changer in NLP? (October 2019), https://towardsdatascience.com/7cca943cf3ad
17. Stewart, S., Burns, D. (eds.): W3C Recommendation, chap. WebDriver. W3C (August 2020), https://www.w3.org/TR/webdriver/
18. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, p. 306. Addison-Wesley Longman Publishing Co., Inc., USA (2005)
19. Vig, J.: Deconstructing BERT, Part 2: Visualizing the Inner Workings of Attention (January 2019), https://towardsdatascience.com/60a16d86b5c1
20. Wang, W., Liu, M., Zhang, Y., Xiang, J., Mao, R.: Financial numeral classification model based on BERT. In: Kato, M.P., Liu, Y., Kando, N., Clarke, C.L.A. (eds.) NII Testbeds and Community for Information Access Research - 14th International Conference, NTCIR 2019, Tokyo, Japan, June 10-13, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11966, pp. 193–204. Springer (2019). https://doi.org/10.1007/978-3-030-36805-0_15