Classification of pre-service science teachers' reflections

Project Report by Hugo Robalino (799951) and Lisa Becker (775242)

Project in cooperation with the Physics Education Research, Institute of Astronomy and Physics of the University Potsdam. Part of the Project Module "Text Mining Applications", taught by Prof. Dr. Manfred Stede at University of Potsdam.
Submitted October 7th, 2019.

1 Abstract

The Physics Education Research of the University of Potsdam uses self-written reflections to evaluate the performance of pre-service science teachers, which used to be evaluated by hand by the professors. Machine learning driven classification has proven to be a faster and sometimes more accurate approach than manual classification. The goal of this project was to find out whether natural language processing can be used to automate the process of analyzing those reports and giving the pre-service science teachers feedback on their enactment. Different classification approaches were tested. The highest result 76,32% was achieved with the Random Forest Classifier with the positioning of sentences in a given document as the only feature, which beat the manual inter-annotator accuracy of 72%.

2 Data and Tools

The data was provided by the Physics Education Research of the University of Potsdam. Eighteen pre-service physics teachers provided each up to six written reflections on their teaching enactments that they made during their internship where they prepared and held teaching sessions to classes of grade eight to twelve. They were all provided the same instructions, following the ALACT reflection model[1] and all reports were written in German. The average document length was 875 words. Informed consent was assured with the pre-service teachers, their data was anonymized and no additional requirement for attaining an ethics review was necessary.

Various packages[2] were used to process the data, implement the features and building the classifiers: Numpy (Oliphant, 2006), SpaCy (Honnibal and Johnson, 2015) and Pandas (McKinney et al., 2010) were used mainly for processing the data. The stemming package and various stopword lists were used from NLTK (Bird et al., 2009), classifier packages were obtained from Scikit-learn (Pedregosa et al., 2011). Keyword extraction was performed by the rake-package from multirake and verb tenses were extracted with the RF Tagger.

Finally, it is worth mentioning this project was done using Jupyter Notebook and Python 3.0.

2.1 Preprocessing

The labelling of the paragraphs was provided by two research assistants. They were provided a thorough manual on how to process the raw documents. Based on their annotations, the data was labeled the following way:

Circumstances	31,69%
Description	22,84%
Judgement	23,39%
Alternatives	11,91%
Consequences	8,20%
Irrelevant	1,97%

Since the provided data was scarce (915 instances), we decided to divide the paragraphs into individual sentences and assigned those the same label as their superordinate paragraph. This way we had a total of 2596 instances.

[1] ALACT reflection model: (1) action, (2) looking back on the action, (3) awareness of essential aspects, (4) creating alternative methods of action, and (5) trial (last accessed September 15th, 2019)

[2] Packages: Multirake (last accessed September 15th, 2019)
RF Tagger (last accessed September 15th, 2019)

2.2 Models

We implemented a variety of Machine Learning learning methods in order to compare them to another: Naive Bayes, Linear Support Vector Machine, Logistic Regression and Random Forest Classifier. We chose diverse classifiers to test them against each other to evaluate which one fits best to the task. Deep learning approaches were out of question due to the scarcity of the data.

2.2.1 Naive Bayes

More specifically MultinomialNB, which is a simple classification algorithm, which provides us with a basic base line of the simplest classifier. Also, MultinomialNB is suitable for this task, since we face a multiclass classification problem. Moreover, this classifier is known to work well with discrete features as well as non-discrete features, such as the ones given by TF-IDF.

2.2.2 Linear Support Vector Machine

In this case, the package SGDClassifier was chosen, since it works well with sparse vectors, such as word counts. This classifiers performs a regularized stochastic gradient descent in order to find the minimal loss. The loss was set up to the hinge loss and an L2 regularizer was used, hence resulting in a linear support vector machine. Moreover, we chose not to shuffle the data, since it might provide useful information as explained in the Evaluation Method section.

2.2.3 Logistic Regression

The LogisticRegression package was used in this case. It was decided to also use this classifier, since it has a different approach to classification, more specifically, it provides probabilities of belonging to a certain class. Moreover, it is used with discrete classes, such as the ones in this task. Finally, the solver for optimization was set to 'newton-cg', because it can handle multi-class classification.

2.2.4 Random Forest Classifier

The package RandomForestClassifier was also used, since it tackles in a different manner the problem at hand. It creates several decision rules (decision trees) that are learned from the training data. The strength of this classifier is that it combines several decision trees to return a single result, which makes the classification more robust, since if one decision tree predicts the wrong class, the others might predict the true class. The number of single decision trees was set up to ten, which is also the default value. Finally, the class weight was set up to 'balanced', since this setup takes into account the class proportion, which is helpful with data with unbalanced classes, such as our data as seen in the PReprocessing section.

2.3 Feature Engineering

We implemented different syntactical, lexical, structural and content-specific features as suggested for text classification (Zheng et al., 2006). We implemented the removal of stop words and punctuation, stemming, lemmatization, part-of-speech tags, hapax legomena ratio, average word length, average sentence length, position of sentence or paragraph in document, verb tense extraction and keyword extraction. The word normalization through stemming, lemmatization, part-of-speech tags, verb tenses and keyword extraction was counted and evaluated in a bag of words for each classifier.

It is also worth mentioning, the approach Doc2Vec was implemented, since it also preserves the order of words in the documents (Syntax) and also the order of documents (how paragraphs and sentences follow each other), which might improve classification. In a more detailed way each instance (either paragraph or sentence) is transformed into a vector (50 dimensions in this case), and then put into the vector space where it is easier to analyze how similar and related each document is to each other. Nevertheless an accuracy of less than 40% was obtained, probably because there were few instances, because Doc2Vec approach works better with a large enough amount of data. Therefore, we did no further usage of that method.

The removal of stopwords seemed to worsen the accuracy (by about 5% on average), depending on the collection of stopwords used. Removal of punctuation seemed to have very little or no influence on the accuracy. Thus we did not investigate those two features further.

Stemming and lemmatization are forms of text normalization. Stemming is the process of removing the inflected pre- or suffix such that only the stem of a word remains (i.e. ”walk” and ”walked” would count as the same word when counted). We implemented and tested two different stemmers (Snowball and Cistem, both by NLTK) developed for German, which alternately differed in their performance evaluated by the classifiers by about 2% on average. Lemmatization is closely related to stemming, but goes a step further to the basis of the word. For example ”better” would be reduced to ”good” in lemmatization, but would stay as ”better” in stemming. Word normalization is usually a superior feature over the count of raw words since the count doesn't take morphology into account, which groups a word with all its inflections into one counted item.

Part-of-speech (=POS) tags assign a label to each word which represents its morphological information, which is commonly used for building parse trees. For example, every inflected main or copular verb (like ”walked”) gets the tag ”VAFIN” assigned. A POS tagger was learned based on a SpaCy package working with the TIGER corpus and Wikipedia. POS taggers for German don't catch information regarding tempus, which is why we additionally implemented a verb tense extractor.

Concerning verb text extraction, the RFTagger was implemented, since it provides more detailed tags, which gives us more information about morphology, such as gender, case and most importantly tense. This is specially suitable due to the rich morphology found in German (Helmut and Laws, 2008). A feature representation of texts (which was named 'Verb Tense') was then made by using RFTagger, which was then used with different classifiers.

As for key word extraction, RAKE was implemented, which stands for Rapid Automatic Keyword Extraction. RAKE extracts the most important topics (words in this case) by first removing stop words, then parsing the string of text into candidate words and finally giving those candidate words a score in computing their degree, which is how often they co-occur with other candidate words, and calculating the ratio between the degree of a word and its frequency. After such process, words with the highest score are selected as key words (Michael W. Berry, 2010). The reason behind choosing key word extraction as a feature representation for the data is they carry more meaning and can better represent the text while reducing the feature space since there are fewer words.

Finally, the raw text was also used as a basic feature representation, which was then used for the word counts. In summary, the following features were used for the classification: Raw text, Lemmas, Stems (Cistem), Stems (Snowball), POS tags, Verb Tense and Keywords.

2.3.1 Position in Text

The position of the paragraphs in a document was also given in the data provided by the Physics Education Research of the University of Potsdam as two features: 'Begin' and 'End', which indicates the number of the line where each paragraph begins and ends. As stated in the abstract, an accuracy of 76,23% was achieved by using only this two features for classifying sentences (instead of paragraphs) and the Random Forest Classifier. It is nevertheless paramount to point out this accuracy was achieved with the first version of the data, which had 223 paragraphs in total. However, when the last version of the data was used (915 paragraphs), the accuracy seemed to worsen to 58,24% (using the positioning of sentences and the Random Forest Classifier).

It is also relevant to note the data was shuffled for training and testing this feature, as opposed to the evaluation method used for training and testing the other features.

Finally, it was decided to test this feature alone, without combining it with other text features (such as raw text or lemmas), since only two values are provided and their meaning would be lost when combined to other text features when they are converted in vectors of more than 15 thousand dimensions.

2.4 Training Method

In order to test different combinations of features, apart from testing the data in sentences and paragraphs, we took two approaches for representing such features, which we named 'appended' and 'vectorized' respectively. On the one hand, appended sentences/paragraphs means each feature (which is a string) is first combined ('appended') to the others and then that combination of strings is vectorized. On the other hand, vectorized sentences/paragraphs means each individual feature is first vectorized and then features are combined in that vectorized form. In other words, in the 'append' method, features are first appended and then vectorized while in the 'vector' method features are first vectorized and then appended. This in total makes up 4 possible feature representations: vectorized sentences, appended sentences, vectorized paragraphs and appended paragraphs.

Moreover, TF-IDF (Text Frequency - Inverse Document Frequency) was also applied to all feature representations for training the classifiers, since it provides information on how relevant a word is to a paragraph/sentence in the data set. That is to say, paragraphs/sentences with similar relevant words will have similar vector representations as well. Thus, making the feature representation more faithful to the content and therefore improving classification.

For the training of classifiers, a process of feature reduction was undertaken for each classifier and for each possible feature representation (as explained above) making up a total of 16 loops (4 classifiers by 4 feature representations): First, all 7 features were combined and evaluated (according to the method in the following section), then one single featured at a time was dropped and the combination of the remaining 6 was evaluated. The 6 feature combination with the best performance was then kept. We performed this process until only one feature was left. The process was not stopped when there was a decrease in performance by dropping one feature, since an increase in performance could also occur after dropping one or more features.

2.5 Evaluation Method

Cross validation was applied to test the effectiveness. The main reason for choosing this method was the scarce amount of data (915 paragraphs). This way, a classifier is trained with the training data set while set of instances (either sentence or whole paragraph), the developmental test set, is held out to then test if the classifier assigns the correct paragraph to it. After repeating this for every instance, the mean accuracy can be calculated over the results of all trials.

In a more detailed way, cross validation was applied by dividing the data into 3 sets without shuffling, since the ordered paragraphs might also contain important information for the classification as they form a whole document.

As a final step, the confusion matrix of the best combination of classifier, feature combination and feature representation was also done, in order to analyze how each category is classified.

2.6 Discussion

Removal of stopwords, punctuation as well as the implementation of hapax legomena ratio, the average word length and the average sentence length have not proven to be useful features. The last three features are more used for authorship detection, thus we suspect them to be more individual to the author than to the type of paragraph.

The question remains why the accuracy seems to be worse for the counts of lemmas compared to the counts of raw words in all classifiers, as well as combinations of appended and vectorized sentences and paragraphs. This is especially peculiar since the rich morphology in the German language should cause a big difference in the amount of lemmas compared to the raw word counts.

A possible explanation for some features to not achieve the results that we expected is that most packages are optimized for English, not for German. This might result in worse performance, for example regarding part-of-speech tags.

Concerning the best result as shown in Table 1, Logistic Regression with vectorized paragraphs

and Raw text, Lemmas, Stems (Cistem) and Verb Tense as features, it can be said that Stems (Snowball), POS tags and Keywords did not help in achieving a better accuracy while other features seemed to improve it. For instance, the Verb Tense feature was used, which shows that such feature carries relevant information about the paragraph class. The fact that Logistic Regression achieved the best performance might also be explained by the way this method works well with discrete classes, such as the ones in the data.

Furthermore, looking at the confusion matrix at Table 2, it is noticeable how paragraphs belonging to proportionally smaller classes: Alternatives, Consequences and Irrelevant (which make up 20% of the data while being half of the classes) are more prone to be misclassified. A better performance could be achieved by having more balanced classes or deleting the smallest classes.

The quality of future data collection might be improvable on several levels: clearer instructions about the ALACT model and writing the report might improve the quality of the raw data. We found that students sometimes confound elements of reflection (e.g., judging while describing the situation). This limits both the capability of NLP classifiers and pre-service science teachers' ability to cogently reflect on their teaching enactments. Furthermore, inter-annotator agreement was only 72% (assessed through Cohen's κ). It might be possible to improve the quality of annotation by clearer instructions to the annotators or having the data annotated by a single person instead or even having the pre-service teachers having annotate their own data. Besides the quality of the data, an increase in data quantity might also improve the performance of the classifiers, since a small sample size increases the risk for problems that revolve around high variance and it becomes more difficult to avoid overfitting.

3 Conclusion

This first approach proved that automatization of feedback to students is possible to a certain extend. We were able to beat the manual inter-annotator accuracy of 72% with the tools of machine learning, since an accuracy of 76,32% was achieved with the first version of the data. Further improvement of the accuracy might be achieved by the collection and annotation of more input data and the improvement of feature and classifier implementation. The Physics Education Research of the University of Potsdam plans on implementing and improving this classifier for their research to eventually automatize and optimize the feedback process.

References

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

Schmid Helmut and Florian Laws. Estimation of conditional probabilities with decision trees and an application to fine-grained pos tagging. *Coling 2008 Organizing Committee: Manchester, UK*, 57(3): 777–784, 2008.

Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL https://aclweb.org/anthology/D/D15/D15-1162.

Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

Jacob Kogan Michael W. Berry. *Text Mining, Applications and Theory*. Wiley, 2010. ISBN 978-0-470-74982-1.

Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

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

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American Society for Information Science and Technology*, 57(3): 378–393, 2006.

4 Tables

	Vectorized sentences		Appended sentences		**Vectorized paragraphs**		Appended paragraphs	
	Features	CVA[3]	Features	CVA	Features	CVA	Features	CVA
Naive Bayes	Raw Text, Stems (Cistem), POS Tags, Verb Tense	54,55	Raw Text, Stems (Cistem), POS Tags, Verb Tense	55,89	Raw Text, Stems (Cistem)	65,04	Raw Text, Stems (Cistem)	64,59
Linear Support Vector Machine	Raw Text, Lemmas, Stems (Cistem), POS Tags, Keywords	53,66	Stems (Cistem), POS Tags, Verb Tense, Keywords	54,51	Raw Text, Lemmas, POS Tags, Verb Tense, Keywords	66,79	Stems (Snowball), Verb Tense, Keywords	66,34
Logistic Regression	Raw Text, Lemmas, Stems (Cistem), Stems (Snowball), Verb Tense	55,94	Raw Text, Stems (Snowball), Verb Tense	56,20	**Raw Text, Lemmas, Stems (Cistem), Verb Tense**	**69,85**	Raw Text, POS tags, Verb Tense	67,21
Random Forest Classifier	Raw Text, Lemmas, Stems (Snowball), POS Tags, Verb Tense, Keywords	45,07	Raw Text, Lemmas, Stems (Cistem), Stems (Snowball), POS Tags, Verb Tense	44,22	Raw Text, Lemmas, Stems (Cistem), Stems (Snowball), Verb Tense, Keywords	54,57	Raw Text, Lemmas, Stems (Snowball), POS Tags	52,46

Table 1: Best combination of features with cross validation accuracy. [3]CVA = Cross Validation Accuracy in %

	Circumstances	Description	Judgement	Alternatives	Consequences	Irrelevant
Circumstances	**71**	8	3	4	0	0
Description	9	**46**	9	0	0	0
Judgement	12	10	**58**	1	1	0
Alternatives	8	3	3	**20**	4	0
Consequences	8	1	2	4	**11**	0
Irrelevant	3	0	1	0	0	**2**

Table 2: Confusion Matrix Logistic Regression with vectorized Paragraphs