JokeMeter at SemEval-2020 Task 7: Convolutional humor

Martin Docekal, Martin Fajcik, Josef Jon, Pavel Smrz
Brno University of Technology, Faculty of Information Technology
612 66 Brno, Czech Republic
{idocekal,ifajcik,ijon,smrz}@fit.vutbr.cz

Abstract

This paper describes our system that was designed for Humor evaluation within the SemEval-2020 Task 7. The system is based on convolutional neural network architecture. We investigate the system on the official dataset, and we provide more insight to model itself to see how the learned inner features look.

1 Introduction

This paper deals with estimating the humor of edited English news headlines (Hossain et al., 2020b; Hossain et al., 2020a). The illustration of tasks is in Figure 1. The original text sequence is given, which represents a title, with the annotated part that is edited along with the edit itself. Our responsibility is to determine how funny this change is in the range from 0 to 3 (inclusive). This is called Sub-Task 1. We also participate in the Sub-Task 2, in that we should decide which from both given edits is the funnier one. For the second task, we used the approach of reusing the model from the first task as it is described in section 4. So we are focusing the description on Sub-Task 1.

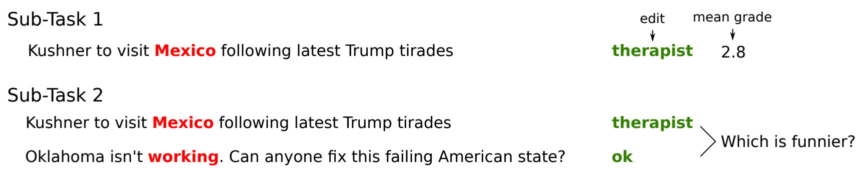
Sub-Task 1
edit
mean grade
Kushner to visit Mexico following latest Trump tirades
therapist
2.8
Sub-Task 2
Kushner to visit Mexico following latest Trump tirades
therapist
Oklahoma isn't working. Can anyone fix this failing American state?
ok
Which is funnier?

Figure 1: Examples for both tasks. The red color in the original title marks part that is substituted with a green word.

Official results were achieved with a Convolutional Neural Networks (CNNs) (LeCun et al., 1999; Fukushima and Miyake, 1982), but we also tested numerous other approaches such as SVM (Cortes and Vapnik, 1995) and pre-trained transformer model (Vaswani et al., 2017).

The humor is a very subjective phenomenon, as can be seen from the inter-agreement on label annotation in Sub-Task 1 dataset[1]. The given data labels do not allow us to learn a sense of humor of a human annotator because the dataset does not specify from whom the grade comes. So, for example, if we have one annotator that likes dark humor and all the others not, we will be considering such a replacement as not humorous no meter if it is excellent dark humor or not. In other words, we may say that we are searching for some most common kind of humor.

The dominant theory of humor is the Incongruity Theory (Morreall, 2016). It says that we are finding humor in perceiving something unexpected (incongruous) that violates expectations that were set up by the joke. There are samples, in the provided dataset, that uses the incongruity to create humor. Moreover, according to Hossain et al. (2020a), we can see a positive influence of incongruity on systems results for the dataset.

[1]The inter-annotator agreement measured with Krippendorff's interval metric is just 0.2 (Hossain et al., 2019).

position	1	2	3	4	5
RMSE	1.179	0.583	0.403	0.63	0.903

Table 1: Results on the Sub-Task 1 train set of the oracle classifier, which always correctly predict the grade on the n-th position.

grade	0	1	2	3
RMSE	1.103	0.587	1.214	2.145

Table 2: This table shows results for a classifier that would always predict the same grade for the Sub-Task 1 train set.

2 Related Work

Computational humor is usually divided into two groups: recognition and generation. Humor generation focuses on creating the humor itself, so it can be a system that is able to tell a joke.

Recognition can be a binary (funny or not) classification task, but in our work, we also want to know how funny a sequence is by rating it with a grade in a given interval. Hossain (2019), which introduced the dataset used in this work, tried to create models that can classify if a given edited title is funny or not, so they were training just binary classifiers, in contrast with our regression approach.

Many other works use this binary approach (Cattle and Ma, 2018; Chen and Soo, 2018; Bertero and Fung, 2016; Weller and Seppi, 2019; Purandare and Litman, 2006). They use various models, such as LSTM (Hochreiter and Schmidhuber, 1997), CNN, and BERT (Devlin et al., 2019), to deal with this task.

Our official results were achieved with a CNN model that is inspired by architecture presented in (Zhang and Wallace, 2015). Their architecture is compact to its size (one layer CNN), so it provides the advantage of using less computational resources, in contrast with big models like BERT. Even with the small size, they were able to achieve promising results for the sentence classification task. Also, such a small model allows us to gain better insight into what is going on underneath.

3 Data

In this section, we would like to point out some interesting facts about the used data. We are focusing on the data for Sub-Task 1 because we used the model trained on the Sub-Task 1 for Sub-Task 2 (more in section 4).

For each example, the dataset (Hossain et al., 2019) provides annotation in the form of grades (0,1,2 and 3) of humor assessment in sorted descending order and the mean of these grades. In most cases, there are five grades per dataset sample (sometimes more). In our work, we always use the first five grades. As can be seen from graphs in Figure 2, the dataset is imbalanced, and the high grades are rare. Also, we investigated how the dataset is imbalanced when considering just single n-th grade and omit the others. Though we can say, from the graph on the right side of the very same figure, that there is still imbalance, we can see that mainly the 2. and 3. positions seem to have smaller fluctuance among the number of samples per grade than the other positions.

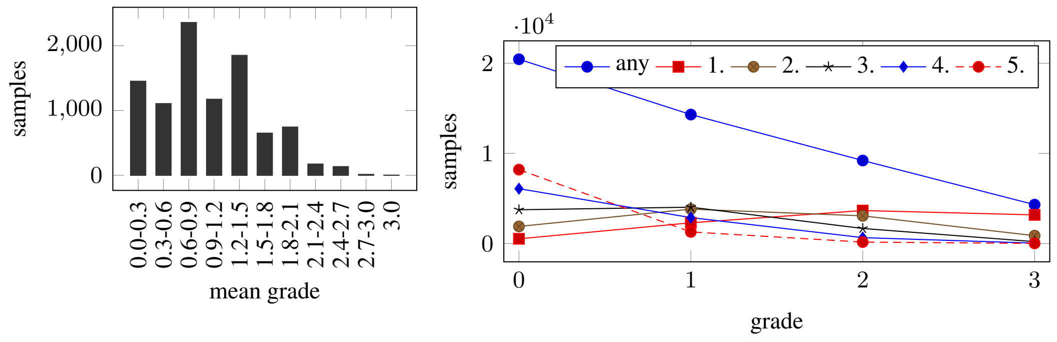

Figure 2: In the histogram on the left side, we can see the number of samples in the train set that have a mean grade in a given bin. The bins are left-inclusive. The graph on the right side shows the number of grades of a given type in the whole train set if we always take just grade on given position per sample.

We also did further analysis to determine prediction quality for the case when we would have an oracle classifier always predicting the grade on the *n-th* position. The results are in Table 1. It can be seen that the third position is superior.

Another thing that we decided to investigate is how the RMSE score would look like if we always predict the same grade. The results are in Table 2.

4 System Description

The main inspiration for our model architecture (JokeMeter) comes from Zhang and Wallace's (2015) work. The used model has CNN architecture illustrated in Figure 3. Firstly the input sequence is assembled. The edit is inserted into the original title after the part that is being edited. Additionally, we add a slash separator, and the whole original/edit location is delimited with two hashtags. In this way, we were able to add input for the model that has complete information about the original and the edited title. We also include tokens to mark the start and the end of a title. The reason behind these tokens is that we want to encode information into an n-gram, whether it is from the beginning/end of a title or not, and possibly make it easier for the model to learn the setup and punchline humor.

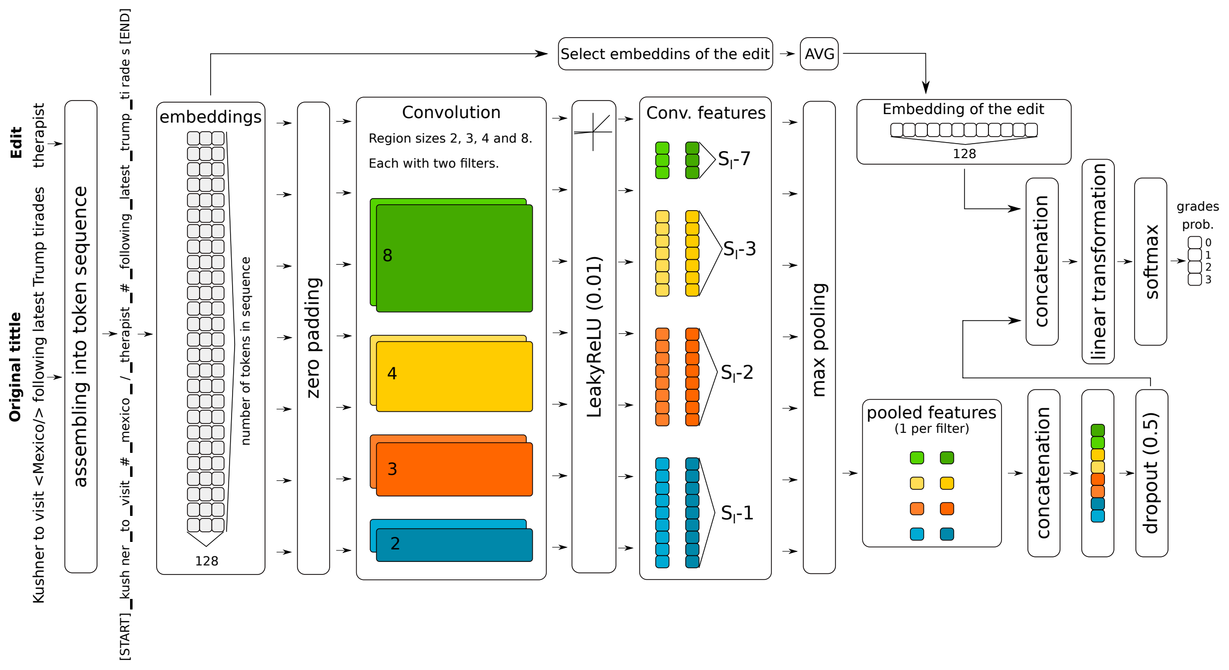

Figure 3: Illustration of our submission model. The S_l is a length of sequence that is processed by the convolution filters.

We tokenized the input by ALBERT (Lan et al., 2019) pre-trained SentencePiece (Kudo and Richardson, 2018) tokenizer. Each token is assigned a 128-dimensional embedding from 30 000 size vocabulary. Right before the convolution, is added zero-padding of size one on both sides of the sequence. Each sequence is 512 tokens long; shorter sequences are padded with unique padding tokens.

We used four different convolution filter region sizes 2, 3, 4, and 8. For each size, we had two filters. These filters are followed with LeakyReLu (Maas et al., 2013) activation (negative slope is 0.01). We also experimented with a model (JokeMeterBoosted) that uses 2048 filters for each size, 2048-dimensional embeddings, and does not use the embedding of the edit.

In the final part of our model, we do max pooling in order to get one feature per filter (8 in total). We concatenate the features to compose a vector, and then that vector is again concatenated with the edit embedding. The edit embedding is an average of embeddings of all tokens the edit is composed of.

On the very end, we perform linear transformation followed by the softmax to get probabilities for each grade. With that configuration, we would not be doing a regression task, so at the test time we must do one final calculation that will transform these probabilities into a grade from the continuous interval [0,3]:

feature	**0**				**1**			
region size	**2**	**3**	**4**	**8**	**2**	**3**	**4**	**8**
σ	0	0.22	0	0.21	0	0.13	0.23	0
$\mathbf{r_s}$	-	-0.43	-	-0.52	-	-0.6	-0.44	-

Table 3: Table of Spearman's rank correlation ($n = 38608$, $p < 0.01$) coefficients between mean grade and each feature. Also, standard deviations of features values.

$$\mathbb{E}[G|S = s] = \sum_{i=0}^{i=3} p_i i\,. \tag{1}$$

Where the G is a grade, s is a input sequence, and p_i is the estimated probability for grade i. In the case of Sub-Task 2, we run the model for both titles separately, and in the end, we made the decision by comparing their estimated grades.

4.1 Convolutional features analysis

We used two filters for each region size because we expected that the model would be able to train one feature that will signalize funny and one feature that will signalize not funny. Nevertheless, our analysis instead shows, it learns features that just signalizes how much not funny the given n-gram is, as can be seen in Figure 4. To gain further insight into this property, we calculated, for the Sub-Task 1 train set, Spearman's rank correlation coefficient between mean grade and each feature (after max pooling). The results in Table 3 shows a negative correlation that corresponds to our hypothesis. An interesting finding is that quite a lot of features have zero variance, which means that constant was learned. That leads us to think that these features reflect the fact that we are able to achieve relatively good RMSE with predicting constant (e.g., one as shown in Table 2) due to the imbalance in the dataset.

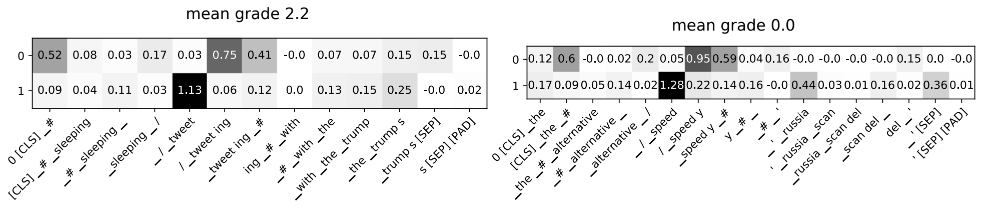

Figure 4: Real word examples of convolved features for trigrams.

5 Experiments

In this section, we describe the experiments we performed, not just with the model that is described in section 4. Apart from models based on neural networks, we compiled several baselines: Decision Tree Classifier (DTC) (Breiman et al., 1984), SVM, k-NN, and Naive Bayes Classifier (NBC). Also, we experimented with a model that uses transformer architecture (ALBERT-base-v2).

The neural models were implemented with PyTorch (Paszke et al., 2019). For the ALBERT and the tokenizer, we used Hugging Face (Wolf et al., 2019) implementations, and for non-neural models, the implementations from the scikit-learn (Pedregosa et al., 2011) were used.

We did two kinds of experiments for all models. The first kind uses all five grades for each sample (*all-grades* training) during the training. Every sample is copied five times, and a grade is assigned to each of them, so we have five samples that have the same content, but each can have a different grade. On the other hand, the second kind of experiment uses only the 3rd grade (*3-grade* training), which has for oracle classifier the best score (see Table 1).

	Sub-Task 1 [RMSE]				Sub-Task 2 [Accuracy]	
	orig. + edit word		edit word		orig. + edit word	edit word
	all grades	3. grade	all grades	3. grade		
DTC	0.987	0.917	0.968	0.914	0.481	0.520
SVM	0.955	**0.834**	0.968	0.914	0.514	0.533
k-nn	0.998	0.860	1.012	0.933	0.542	**0.543**
NBC	1.574	0.911	1.969	1.626	0.482	0.266

Table 4: Results on the test set for non-neural models. The number of neighbors for k-NN differs among the experiments. For the Sub-Task 1 we use $k = 5$, and for Sub-Task 2 $k = 13$ is used.

	Baseline	Best	JM (all)	JM (3.)	JMB (all)	JA (all)	JA (3.)
Sub-Task 1 [RMSE]	0.575	0.497	**0.558**	0.573	0.545	0.533	0.554
Sub-Task 2 [Accuracy]	0.490	0.674	**0.578**	0.538	0.605	0.640	0.615

Table 5: Results on the test set for neural models. We also provide results of the baseline and the best model in the competition. Our official results are in JM (all) column. The JM abbreviation means JokeMeter, JMB is JokeMeterBoosted, JA is JokeALBERT, all means that the *all-grades* training was used and 3. that the *3-grade* training was used.

5.1 Non-neural models

These models deal with classification, meaning the model must decide between 4 grades instead of selecting a value from [0,3]. TF-IDF word features are used for every model. All models are trained on the train set.

We show results for two types of experiments. When the original sequence and the edit (the new word we are inserting into the title) are provided and when we only provide the edit word. The results can be seen in Table 4. These models are not even able to achieve results that can be provided by the simple model that predicts constant (see Table 2). Interestingly, comparable results can be achieved with just using the edit word.

5.2 Neural models

In addition to the model used in our submission (and its version JokeMeterBoosted), we performed experiments with a system using a pre-trained ALBERT model (JokeALBERT). JokeALBERT utilizes contextual embeddings of the whole input sequence from ALBERT and then selects those that belong to the edited word and averages them into one. Finally, the linear transformation with softmax is applied.

To all our neural models, we provided input that uses the same format (see section 4). For both models, we use the cross-entropy loss. We used *Adam* (Kingma and Ba, 2014) with weight decay (Loshchilov and Hutter, 2017) as a optimizer. We stop the training after no improvement in RMSE on the dev set in five subsequent epochs. The results for these models are presented in Table 5. Results for JokeMeter and JokeALBERT were obtained with batch size 16 and learning rate 1e-5.

5.3 Evaluation and baseline

All results for trained models were evaluated with the official scripts. For the Sub-Task 1, we always show the root-mean-square error (RMSE) metric, and for the Sub-Task 2, the accuracy is used.

The baseline system for Sub-Task 1 always predicts the mean funniness grade from the training set (0.936), and for Sub-Task 2, it always predicts the most frequent label in the training set (1).

6 Conclusion

The system description was provided, and we compare the achieved results of the official model with several other models, including the baseline and the best team in the competition.

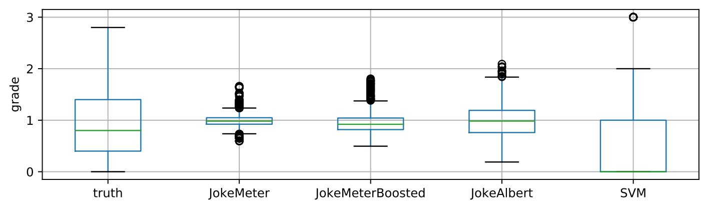

Figure 5: Comparison of grade predictions for multiple models and the truth labels. We can see that JokeMeter and JokeALBERT's predictions are focused on a small interval around the one. JokeMeter, JokeMeterBoosted and JokeALBERT are for all-grades training, and the SVM is for 3-grade training.

In future work, it should be more investigated if imbalanced dataset and small inter-annotator agreement caused that the JokeMeter model was more focused on the prior probabilities of grades and not on the input itself (see Figure 5).

Acknowledgements

This work was supported by the Czech Ministry of Education, Youth and Sports, subprogram INTERCOST, project code: LTC18054.

References

Dario Bertero and Pascale Fung. 2016. A long short-term memory framework for predicting humor in dialogues. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 130–135, San Diego, California, June. Association for Computational Linguistics.

L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. 1984. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Andrew Cattle and Xiaojuan Ma. 2018. Recognizing humour using word associations and humour anchor extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1849–1858, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117, New Orleans, Louisiana, June. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

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

Kunihiko Fukushima and Sei Miyake. 1982. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nabil Hossain, John Krumm, and Michael Gamon. 2019. "president vows to cut <taxes> hair": Dataset and analysis of creative text editing for humorous headlines. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 133–142, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020a. Semeval-2020 Task 7: Assessing humor in edited news headlines. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Nabil Hossain, John Krumm, Tanvir Sajed, and Henry Kautz. 2020b. Stimulating creativity with funlines: A case study of humor generation in headlines. In *Proceedings of ACL 2020, System Demonstrations*, Seattle, Washington, July. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101.*

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

John Morreall. 2016. Philosophy of humor. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 edition.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for F*R*I*E*N*D*S*. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215, Sydney, Australia, July. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Orion Weller and Kevin Seppi. 2019. Humor detection: A transformer gets the last laugh. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3621–3625, Hong Kong, China, November. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification.

A Supplemental Material

All the results presented in this section are averages from 3 runs. The evaluation is always done on the dev set.

A.1 Embeddings

In Figure 6 can be seen the influence of the token embedding size on the RMSE. The rest of the JokeMeter model configuration remains the same as described in section 4.

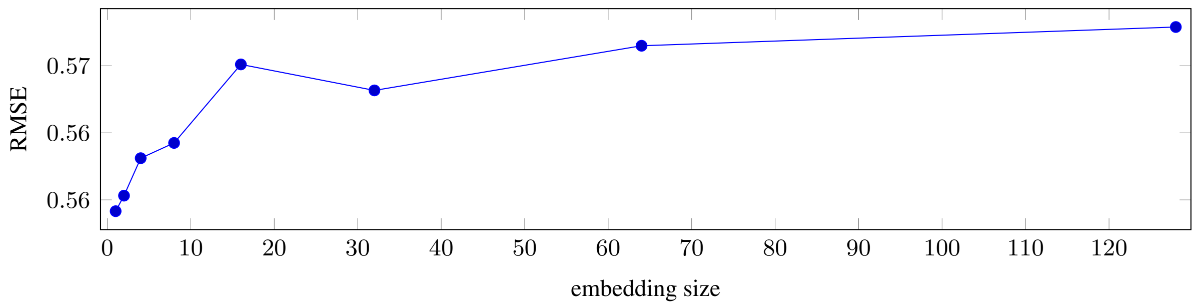

Figure 6: Influence of the size of the token embedding on the RMSE.

A.2 Convolutional features

In Figure 7 can be seen the influence of the used convolutional filters per region size on the RMSE. The rest of the JokeMeter model configuration remains the same as described in section 4.

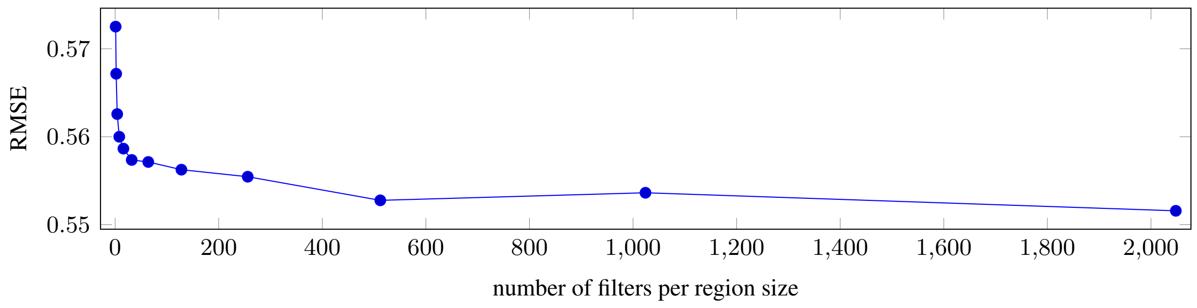

Figure 7: Influence of the number of filters per region size on the RMSE.

As shown in Figure 8 there is a different relation between token embedding size and RMSE for 2048 convolutional filters per region size than for the default 2 (see Figure 6).

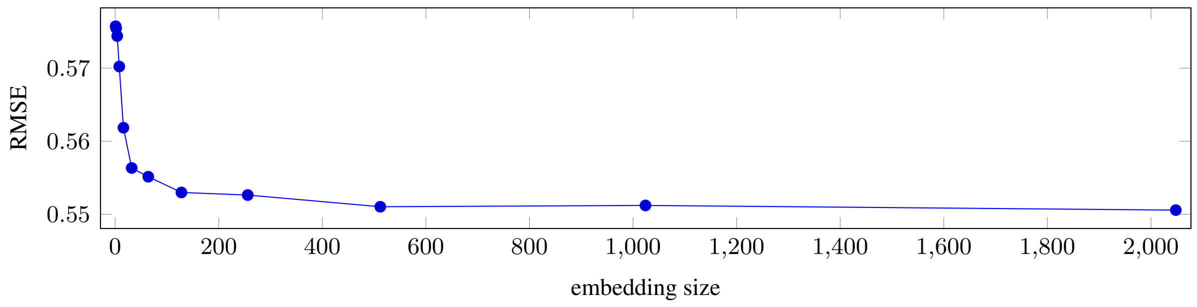

Figure 8: Influence of the size of the token embedding on the RMSE for 2048 convolutional filters per region size.

A.3 Ablation experiments

In this section are presented results for ablation experiments for JokeMeter that uses 2048 convolutional filters per region size and 2048 size of the token embeddings. The rest of the model configuration remains the same as described in section 4.

The influence of used features summarizes Table 6. We can see that the usage of edit embedding does not improve results. We used these findings to create a model that is using 2048 convolutional filters per region size, 2048 dimensional token embeddings, and no edit embedding; we call it JokeMeterBoosted.

	RMSE
convolutional features only	0.550260959621652 $\pm$ 0.0012
edit embedding only	0.63520130408731 $\pm$ 0.0005
convolutional features and edit embedding	0.5505674648279042 $\pm$ 0.0008

Table 6: Results for the ablation experiments on the JokeMeterBoosted.

A.4 Batch size and learning rate analysis for JokeMeterBoosted

According to Figure 9 we used for the JokeMeterBoosted batch size 64 and learning rate 1E-5.

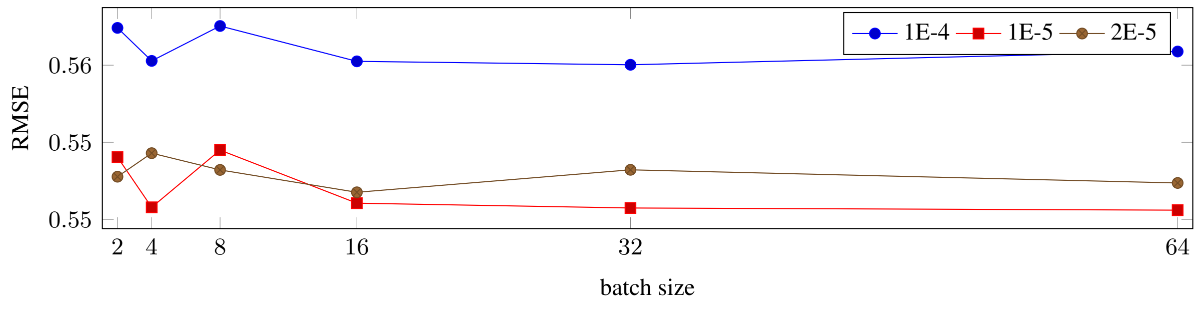

Figure 9: JokeMeterBoosted RMSE for three learning rates depending on the batch size.