Buhscitu at SemEval-2020 Task 7: Assessing Humour in Edited News Headlines using Hand-Crafted Features and Online Knowledge Bases

Kristian Nørgaard Jensen
IT University of Copenhagen
krnj@itu.dk

Nicolaj Filrup Rasmussen
IT University of Copenhagen
nicr@itu.dk

Thai Wang
IT University of Copenhagen
twan@itu.dk

Marco Placenti
IT University of Copenhagen
mapl@itu.dk

Barbara Plank
IT University of Copenhagen
bapl@itu.dk

Abstract

This paper describes our system to assess humour intensity in edited news headlines as part of a participation in the 7th task of SemEval-2020 on "Humor, Emphasis and Sentiment". Various factors need to be accounted for in order to assess the funniness of an edited headline. We propose an architecture that uses hand-crafted features, knowledge bases and a language model to understand humour, and combines them in a regression model. Our system outperforms two baselines. In general, automatic humour assessment remains a difficult task.

1 Introduction

Humour aims at generating amusement and laughter and can for this reason be considered one of the features enabling the creation of relationships in the interactions between humans. Understanding humour requires factual knowledge, context comprehension and—arguably—intelligence. Multiple factors play a role in the definition of humour, such as geographical location, culture, level of education and many others. This obviously makes the task of humour detection very hard for machines and artificially intelligent systems. In recent years, researchers operating in the field of computational linguistics have started to look into the topic, and a lot of progress has been made since the seminal paper by Mihalcea and Strapparava (2005). However, the quality of data sets leaves many questions unanswered, mainly because they are made of single punchlines or because sentences are divided into binary categories. Hossain et al. (2019) makes a remarkable effort on creating a data set of edited headlines where each headline is assigned a score representing the intensity of humour of that headline. This innovative data set enables researchers to conduct studies on a more granular level and may unlock novel techniques to get closer to a more efficient and successful computational model of humour. In this paper we propose an architecture that accounts for multiple factors that we believe play an important role in detecting the intensity of humour in a headline. In order to analyse the sentences, we include hand-crafted features extracted from the sentence itself and enable the system to look for the meaning of unknown objects using NELL (Never-Ending Language Learning) (Mitchell et al., 2015). This paper is a description of a system providing a solution to the SemEval2020 Task 7 (Hossain et al., 2020), which ranked 22nd out of 49 teams.

The task is comprised of two sub-tasks. The first task is a regression task aimed at predicting the humour intensity of an edited headline. The second one is a classification task in which it is required to select the funnier headline out of the two provided. Our main focus was on the first sub-task, as predicting the humour intensity of the two headlines would imply establishing which of the two has the higher score.

2 Related Work

Due to its complexity and the prerequisite for a deep understanding of humour, previous research contributions towards automatic humour recognition have been made in selected aspects of humour. Mihalcea and Strapparava (2005) introduced a binary classification task in humour recognition, using humour-specific features such as alliteration, antonyms and adult slang in conjunction with traditional

text classification models: Naive Bayes and SVM. Due to feasibility, the work focused solely on short sentences, one-liners, news headlines and proverbs. In recent years with the emergence of Deep Learning, the usage of Convolutional Neural Networks and Highway Networks for humour classification tasks with a similar scope focused on puns, one-liners and short jokes was presented in Chen and Soo (2018). Another way to find jokes is on different social media platforms. Weller and Seppi (2019) used data scraped from Reddit to assess whether a joke is funny or not. They demonstrated the effectiveness of the transformer architecture for humour classification. Purandare and Litman (2006) classified the spoken turns of the TV-show FRIENDS into humour and non-humour classes. They did so by employing the ADTree algorithm on lexical, prosody and speaker features.

Another related work dedicated to the effort to assimilate external knowledge is the study reported in Yang and Mitchell (2017). The work introduced an approach to leverage external knowledge bases, such as NELL (Mitchell et al., 2015) and WordNet (Miller, 1995) (a lexical database), in order to integrate the background knowledge and enhance the learning on LSTM.

Finally, with the extensive work done in Hossain et al. (2019) on humour generation, the goal of the study was to generate a carefully curated dataset of news headlines with simple edits, based on robust generation strategies that emphasise free form over traditional jokes with a strong template. This facilitates further research into the shared tasks described and performed in this report.

3 Data Analysis

The data set consists of micro-edits on headlines: one word has been replaced by another word, e.g. "How Trump Just Made ~~America~~ (Pilates) Less Safe". Five Mechanical Turks are asked to assign a score between $0-3$ to each headline (0: not funny, 1: slightly funny, 2: moderately funny, to 3: funny) (Hossain et al., 2019). The overall score of the headline is then the average of those five scores. Similarly to Hossain et al. (2019), we find the scores to have a correlation with the headline length - measured as number of tokens present in it - and the relative position of the replaced word within the headline. The humour increases if the edit happens toward the end of the headline, as can be seen in Figure 1.

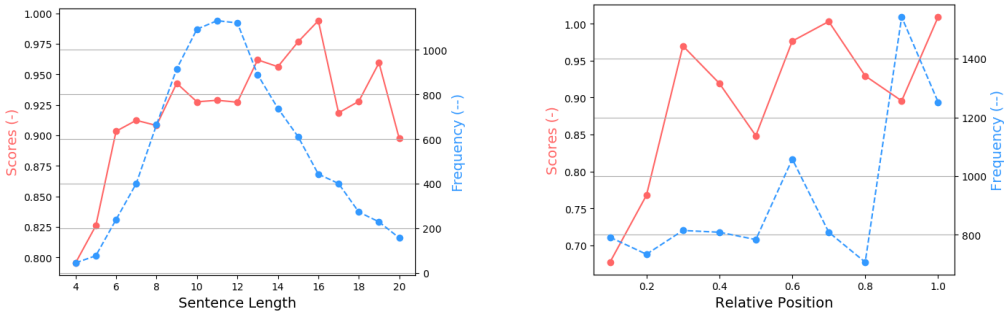

Figure 1: Distributions of headline length and edit relative position

4 System Description

In this section we outline the structure of our system and go into details on the different components. The proposed system consists of three encoders which handle three different types of inputs. Sections 4.1 to 4.3 explain the inputs and how they are handled in each of the three encoders. Section 4.4 outlines how the results from each of the three encoders are combined and processed. Section 4.5 goes through the training parameters and extra tricks we use to get more performance from our model. The overall structure of the model is shown in Figure 2. Section 4.6 explores further improvements we have developed after the official submission deadline.

4.1 Word Encoder

The word encoder handles representations of both the replaced and the replacement words in the edited headline. The encoder first encodes each of the words using a pre-trained neural probabilistic language model (NNLM) (Bengio et al., 2003). For each of the two words it processes the representation using a Feed Forward Neural Network (FFNN) that consists of three layers (See appendix A). The NNLM and

the FFNN weights are the same for each of the two words, and thus it works as a simple Siamese network (Chopra et al., 2005). After both of the words have been processed the representations are concatenated before proceeding in the neural network.

4.2 Feature Encoder

The feature encoder takes four features that encode humour specific information from the headlines. Each feature helps the model to better understand the concepts behind humour and helps outline the strategies used by the annotators. The features are processed using a 2 layer FFNN (See appendix A).

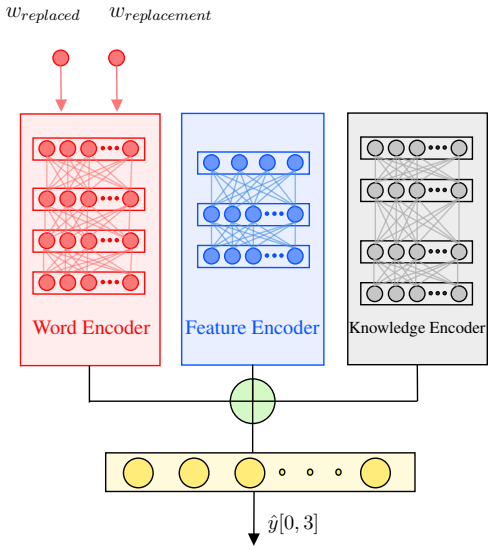

Figure 2: Our proposed model architecture

Relative Position The first feature encodes the relative position of the replaced word. The position index is normalised by the maximum index to provide a number between $0 - 1$. It informs the system of whether the headline functions as a punchline or not.

Sentence Length The second feature encodes the length of the headline, as shown in fig. 1. The length is normalised by the maximum length in the data set, thus providing a number between $0 - 1$. Hossain et al. (2019) uncovered a relation between the length of the headline and the score, showing that the longer headlines had the possibility of also scoring higher. This makes it a promising feature to include.

Phonetic Distance For the third feature the replacement and the replaced words are transcribed into phonemes and the Levenshtein distance between them is calculated, as shown in table 1. The distance is normalised by the maximum phoneme length. This feature is used to encode information regarding the strategy uncovered by Hossain et al. (2019), about connections between the replaced and the replacement word. Here the annotators often replaced a word with either a similar sounding word or a semantically different word.

Relative Distance The fourth and last feature encodes the cosine distance between the replaced and replacement word embeddings. FastText embeddings trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news data (Mikolov et al., 2018) are used. Another of the strategies found by Hossain et al. (2019) is the insertion of incongruity. We hypothesise that finding the similarity between the two words (replaced and replacement) is to some degree related to incongruity.

Replaced word	Replacement word	Levenshtein distance
'Syria' → 'S IH1 R IY0 AH0'	'cereal' → 'S IH1 R IY0 AH0 L'	0.1176
'coup' → 'K UW1'	'ignorance' → 'IH1 G N ER0 AH0 N S'	0.9474

Table 1: Example of phonetic distance feature showing transcription from grapheme to phoneme.

4.3 Knowledge Encoder

The knowledge encoder is searching the headline for any known entities occurring in the NELL database or hypernym in WordNet. Table 3 lists some example headlines that contain entities such as named entities that we have highlighted in blue, which we believe would benefit from relations and its implication through their common parent defined by NELL. In contrast to a lexical database, NELL features entities that are obtained by reading the web, thus filling the gap in comprehension of concepts that are time- and event based. Even though NELL is a large network, it alone is insufficient in covering a significant part of each headline (see table 2). In order to expand said coverage, nouns are extracted from WordNet excluding entities present in NELL.

	Min	Average	Median	Max
NELL	0	4.8	5	13
NELL + WordNet	1	7.2	7	18

Table 2: Number of entities found in training data and NELL dictionary

Each noun is converted to an IS-A relation by adding its first occurring hypernym as its generalisation. With the integration of WordNet into NELL, our coverage of entities in each headline improves significantly (see table 2).

For each entity in the NELL-WordNet vocabulary, we have created an embedded representation using a Neural Association Model (NAM) presented by Liu et al. (2016)[1]. The model looks up each word in the headline and checks for a occurrence in the NELL-WordNet vocabulary. If it does exist, it will be represented by the corresponding embedding, and if it does not exist it is represented by a zero vector. The found entity embeddings and zero vectors are then summed together before they are processed in a 2-layer FFNN (See appendix A).

Original Headline	Substitute
Breitbart News 29th Most Trafficked Site in America , *overtakes* **PornHub** and **ESPN**.	combines
Barack Obama threatens to upstage **Donald Trump** 's *Europe* trip as he visits **Germany**.	acid
Delhi *smog* chokes **India** capital with air pollution 10 times worse than **Beijing**.	curry
Elon Musk has just blasted the world 's most powerful rocket into *space*.	wall

Table 3: Example of headlines from the training data that would not have turned out as fun without the necessary background knowledge. Italic (red) denotes a replaced word, and bold (blue) denotes a named entity that would benefit from the integration of a knowledge base like NELL.

4.4 Output

A simple linear regression is applied to the concatenation of the three encoders output described above. It predicts an output in the range $[0, 3]$. Several output layer configurations were tested but none outperformed this simple regression.

4.5 Experimental setup

We used Keras (Chollet and others, 2015) with the TensorFlow backend (Abadi et al., 2015). The pre-trained models, NNLM[2] and Albert[3], were provided by the TensorFlow Hub module. For the phonetic feature we used the "g2p: English Grapheme To Phoneme Conversion" (Park and Kim, 2019) library.

The Adam optimiser (Kingma and Ba, 2015) was used with a step decay learning rate schedule. The learning rate was initialised to 0.005 and drops by a factor of 2 every 10 epochs. The model used for the official submission was trained for 25 epochs where it converges. For the subsequent hyperparameter tuning we used the newly created Keras-Tuner library, which is built specifically for Keras.

4.6 Improvements after official submission

After the official submission, the model has been further improved in two ways. First, we dedicated time for hyperparameter tuning. The Hyperband optimisation method (Li et al., 2018) was used for hyperparameter tuning. Hyperband is a Bandit-based approach to the hyperparameter tuning problem. The algorithm extends the SuccessiveHalving algorithm by using it as a subroutine. It does so to automatically select the number of configurations to try given a finite budget. The resulting model can be seen in appendix B. The tuning was done over all parameters in the network, and ran for 8 Hyperband iterations. We tested multiple layers in each of the encoders, different layer sizes, the amount of dropout and the

[1]The embeddings can be found in the project repo: `https://github.com/bachelorbois/HumorHeadlines`
[2]NNLM: `https://tfhub.dev/google/tf2-preview/nnlm-en-dim128/1`
[3]Albert: `https://tfhub.dev/tensorflow/albert_en_base/1`

activation function. It was found that adding extra layers to the output layer did not result in an increase in performance, thus the output was kept as is. The resulting score can be seen in table 4.

In the official model configuration only the word to be replaced and the replacement word is used as context of the headline itself. The original idea was to use the NNLM part of the word encoder to encode the entire sentence. However, it was found during preliminary experiments that this did not improve performance compared to encoding just the words. In order to address this an extended configuration is made with a separate context encoder based on an Albert model (Lan et al., 2019). The encoder takes the entire headline except the replaced word and creates context embeddings for it. The contextual embeddings are created by running the headline through the Albert model and extracting the pooled output. The embeddings created by the Albert model are processed using a 2 layer FFNN to scale down the representation and let the model process it before concatenating it with the other encoder results. The new model architecture can be seen in fig. 4.

5 Results

In this section we outline the results gathered from experiments with both the official model and the alternative model created after the official submission deadline. As main evaluation metric we use Root Mean Square Error (RMSE), which is defined as $\sqrt{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2/n}$ (1).

Official Results The proposed model achieves an RMSE of 0.5511 on the test set. This is also our final and **official** score in the competition. It gives a ranking of 22 out of 49 teams. The official baseline given for the task is the overall mean funniness grade in the training set, as reported in table 4. To create our own baseline we have set up a Linear Regression model that uses our hand-crafted features. As shown in table 4 the linear regression model reaches only around baseline performance. Our official full model presented in this paper outperforms both baselines.

System	Test Score
Official Baseline (Mean)	0.57471
Linear Regression	0.57361
Our official submission	0.55115
HP Tuned official model	0.54376
Model w/ Albert context	0.54341

Table 4: Scores on the test set

Improvements We present two additional runs as introduced in section 4.6. Results on the test set are shown in the two last rows in table 4. We note that both tuning the system on dev and the integration of context from the headline (the context encoder with Albert embeddings) pushes performance further (after the official submission), thereby confirming our hypothesis that more headline context is helpful.

6 Discussion

In this section we show results of an ablation study of our official model and discuss limitations of it.

Ablation Studies Figure 3 shows the performance on the training and development sets for each of the ablated features of the submitted full model. For a more detailed overview see Appendix D.

Word identity of the micro-edits turns out to be the most important feature. A clear decrease in performance (higher RMSE) can be observed on both training and development set when the Word Encoder (WE) is removed. Likewise, removing one of the two word inputs to the Word Encoder causes an increase in RMSE on the training set. Excluding the Knowledge Base (KB) tells a similar story, causing an increase of median RMSE on the training set (however, not in mean score, as shown in the appendix).

Unfortunately, our hand-crafted features alone cause no detectable difference on either the training or the development set. Neither the feature encoder nor the knowledge encoder cause an increase in development error when removed individually. Interestingly, when both encoders are removed simultaneously (KF) an increased training error can be observed, albeit the difference is negligible on the development data. When removing the Word Encoder in combination with one of the two other encoders it performs notably worse, as expected. It is interesting to note that the combined word and feature encoder model results in the highest drop (see appendix D), but it is also the most unstable model with the highest variation as the box plot in Figure 3 reveals. This points at the importance of investigating both mean and median scores.

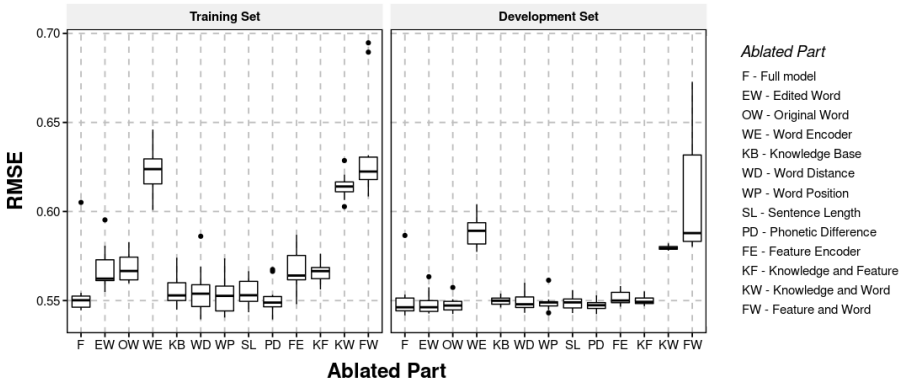

Figure 3: Ablation Study Results. RMSE: Lower is better.

From the single hand-crafted features we notice that contrary to expectations, the phoneme-based feature hurts performance; leaving it out improves overall RMSE, which is disappointing. Similarly, the position or length-related features of the headline itself are not helpful either.

Limitations An underlying assumption of the proposed architecture is that some knowledge of broader context is required in order to accurately understand humour. This is also noted by Hossain et al. (2019), who state that understanding humour often requires real world-knowledge and common sense. Successfully exploiting such knowledge in a neural model is still very difficult.

We propose one way to integrate knowledge-base information via the Knowledge Encoder. However, we are unable to show any significant improvement in model performance by integrating the knowledge in the manner proposed. A possible reason for this is that the NELL and WordNet databases do not encode the necessary information for this particular task. It is also possible that the way it is employed in the model is not appropriate for the type of data it is based on, or our CBOW aggregation is too simplistic.

Another limitation is the set of hand-crafted features. The phoneme-based feature using Levenshtein distance surprisingly hurts performance. Future work could study other ways of leveraging knowledge bases, integrating hand-crafted features and contextualised word representations.

7 Conclusions

We proposed a simple neural model which uses three decoders to model humour intensity of edited headlines. Our official submission obtained an RSME of 0.55115 (top scoring team: 0.49725). Our ablation study shows that the most important information is word identity of the micro-edits, followed by knowledge base representations. However, we note that the way that we implement the knowledge encoder and phonemic information is somewhat ineffective in capturing the information we hoped it would, which leaves room for future work on this challenging task.

Acknowledgements

We would like to thank the HPC support at ITU, especially Frey Alfredsson, for support for the computational resources used in this work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

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.

François Chollet et al. 2015. Keras. `https://keras.io`.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE.

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.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538. Association for Computational Linguistics.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Kyubyong Park and Jongseok Kim. 2019. g2pe. `https://github.com/Kyubyong/g2p`.

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.

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 3612–3616.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446, Vancouver, Canada, July. Association for Computational Linguistics.

A Official Model Parameters

Feature Encoder		Knowledge Encoder		Word Encoder	
Layer 1	16	Layer 1	32	Layer 1	64
Dropout 1	0.5	Dropout 1	0.5	Dropout 1	0.5
Activation 1	relu	Activation 1	relu	Activation 1	relu
Layer 2	16	Layer 2	16	Layer 2	32
Dropout 2	0.5	Dropout 2	0.5	Dropout 2	0.5
Activation 2	relu	Activation 2	relu	Activation 2	relu
				Layer 3	16
				Dropout 3	0.5
				Activation 3	relu
Output	1				

B Hyper Parameter Tuned Model

Feature Encoder		Knowledge Encoder		Word Encoder		Context Encoder	
Layer 1	104	Layer 1	120	Layer 1	480	Layer 1	128
Dropout 1	0.4	Dropout 1	0.05	Dropout 1	0.15	Dropout 1	0.5
Activation 1	relu	Activation 1	sigmoid	Activation 1	relu	Activation 1	sigmoid
Layer 2	8	Layer 2	104	Layer 2	144	Layer 2	160
Dropout 2	0.3	Dropout 2	0.2	Dropout 2	0.5	Dropout 2	0.5
Activation 2	relu	Activation 2	sigmoid	Activation 2	relu	Activation 2	relu
				Layer 3	8		
				Dropout 3	0.2		
				Activation 3	tanh		
Output	1						

C Model with Context

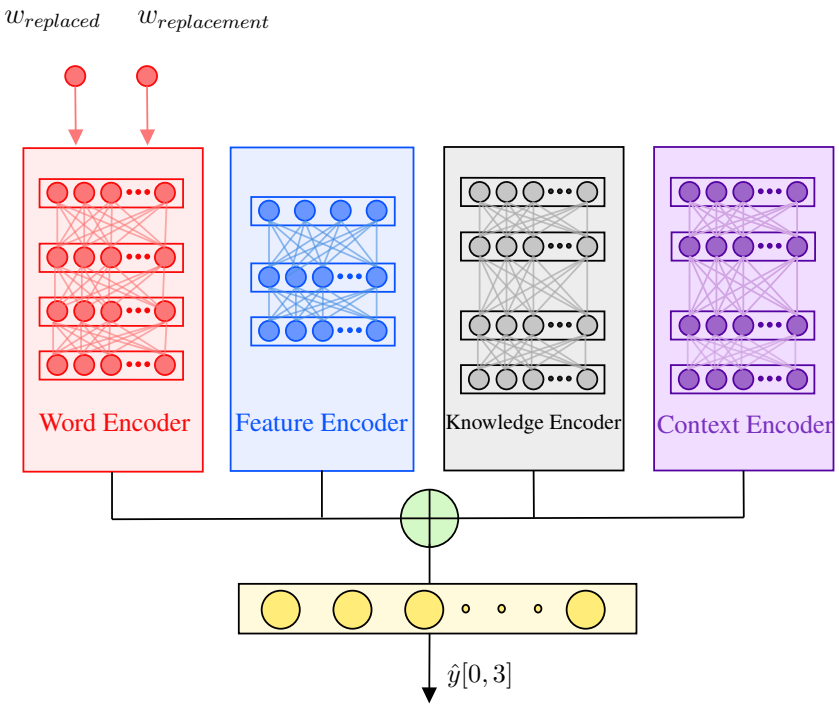

Figure 4: Model architecture with context encoder included

D Ablation Study Results

Configuration	Training			Development		
	Median	Mean	Stddev	Median	Mean	Stddev
F	0.550	0.555	0.0180	0.546	0.551	0.0131
EW	0.562	0.568	0.0121	0.546	0.549	0.00677
OW	0.567	0.568	0.00796	0.547	0.548	0.00425
WE	0.624	0.624	0.0146	0.589	0.589	0.00948
KB	0.553	0.557	0.0102	0.550	0.550	0.00247
WD	0.554	0.555	0.0140	0.548	0.549	0.00504
WP	0.553	0.553	0.0102	0.549	0.549	0.00492
SL	0.553	0.555	0.00780	0.549	0.549	0.00383
PD	0.549	0.551	0.00923	0.547	0.547	0.00348
FE	0.564	0.567	0.0121	0.550	0.551	0.00387
KF	0.567	0.566	0.00556	0.549	0.550	0.00257
KW	0.614	0.614	0.00725	0.579	0.580	0.00119
FW	0.622	0.634	0.0313	0.588	0.607	0.0336

Table 5: Ablation Study Results; F = Full Model; EW = Edited Word; OW = Original Word; WE = Word Encoder; KB = Knowledge-Base Encoder; WD = Words Distance; WP = Words Position; SL = Sentence Length; PD = Phonetic Difference; FE = Feature Encoder; KF = Knowledge and Feature Encoders; KW = Knowledge and Word Encoders; FW = Feature and Word Encoders.