IT UNIVERSITY OF COPENHAGEN

Assessing Humour in Edited News Headlines using Hand-Crafted Features and Online Knowledge Bases

Nicolaj Filrup Rasmussen
nicr@itu.dk

Kristian Nørgaard Jensen
krnj@itu.dk

Marco Placenti
mapl@itu.dk

Thai Wang
twan@itu.dk

Bachelor thesis
BSc. Data Science

May 15, 2020
Supervisor: Barbara Plank

ABSTRACT

In this thesis we develop an architecture aimed at tackling humour intensity prediction. The task has a continuous label space contrary to much previous work, which has mostly concerned itself with discrete (and often binary) classification.

Using a combination of techniques the regression model seeks to incorporate many aspects of humour. By combining modern neural encoders with classical hand-crafted features and neural language models we hypothesise that it is possible to capture many perspectives of the complex task.

By comparing a variety of configurations to relevant baselines we conclude that the proposed model performs well. An ablation study shows that the main contributor to the models success is the neural language model. By analysing the components further the work seeks to explore why this is, and proposes some possible answers for why the components underperform, and how this can be addressed in future work.

CONTENTS

LIST OF FIGURES

PREFACE

We are very happy and honoured to have participated in SemEval 2020 along with some brilliant researchers from around the world. And we are thrilled that this task has helped guide us, as we have written this thesis.

We would like to thank Barbara Plank, our supervisor. You helped us find the task in the first place and you have helped us find a way of making it our own. You always believed we were capable and were always able to get our work on the right track and keep our heads in the game.

This thesis has been created under some very strange circumstances. Due to the ongoing COVID-19 pandemic we have seen the work done from home, apart and in a strange and unknown environment. We have struggled with motivation and found it hard to work, but have managed to see it through in no small part due to our joint commitment to have a culmination to our degree that we can be proud of.

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. Due to the wide variety of humour the amount of background knowledge varies from joke to joke, but for most one needs to have a basic understanding of the context. Some jokes are build up with an introduction of the context and the needed knowledge whilst others builds upon a mutual understanding of a given situation. For the latter to work there are multiple factors that play a role in the definition of humour, such as geographical location, culture, level of education and many others. All these factors makes the task of humour detection very hard for machines and artificially intelligent systems in general. However, it is equally difficult for humans to classify what is humour and what is not, purely based on text. Jokes often build upon a special construction, pronunciations, or sarcasm which are all hard to notice from the written words.

Another property of humour that is making it difficult to work with is the fact that humour is not a binary property of some text or telling. Jokes and stories can have varying degree of humour, meaning that not all jokes are equally funny neither are they either funny or not funny. Moreover, based on the amount of background knowledge needed and the recipients knowledge thereof the joke can be perceived with varying degrees of humour intensity.

In recent years, researchers operating in the field of computational linguistics have started to research 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 has left many questions unanswered, mainly because they were made of single punchlines countered by news headlines or because the labels were divided into binary categories thus ignoring the interesting intensity property of humour.

Hossain et al. (2019) made a remarkable effort on creating a data set of edited headlines where each headline is assigned a score representing the intensity of humour that headline was perceived with. This innovative data set enables researchers to conduct studies on a more granular level and might unlock novel techniques to get closer to a more efficient and successful computational model of humour. The data set further distinguishes itself in that contrary to simpler binary data sets, such as the one presented by Mihalcea and Strapparava (2005), it does not need negative (not funny) examples. This enables us, as we will see, to sample the humour from the same distribution in contrast to having purposely made jokes countered by serious texts, thus giving us a homogeneous data set.

In this thesis we develop a basic and an extended neural architecture that accounts for multiple factors that we believe play an important role in detecting the intensity of humour in an altered news headline. The idea to the thesis stems from a shared task given at SemEval 2020 (Hossain et al., 2020). As part of our thesis we participated in the shared task, which produced a system description paper which holds parts of this thesis (Jensen et al., 2020). All of the data and the code used to produce the following results is on Github[1].

[1] Repository: https://github.itu.dk/SentimentGroup/HumorHeadlines

In the thesis we have chosen to limit our work to two research questions. The first of which is:

Research Question 1. Is it possible to use hand crafted features to aid in a humour recognition task?

In order to analyse the sentences, we included hand-crafted features extracted from the sentence itself. We formally hypothesise that:

Hypothesis 1.1. By encoding meta information like sentence length, it is possible to inject useful information into a humour recognition model.

Moreover, humour comes in a variety of compositions and we believe that we are able to capture information about these varieties in similar hand-crafted features. Our second hypothesis for RQ 1 is as follows:

Hypothesis 1.2. Using hand-crafted features can help in capturing structures of varying kinds of humour, thus allowing the model to generalise better.

As we described in the beginning of this chapter, understanding humour requires both factual and contextual knowledge. Our second research questions is concerned with this property of humour and asks the following:

Research Question 2. Is it possible to enhance humour recognition systems by using Knowledge bases in order to encode domain knowledge?

We enabled both of our systems to look for the meaning of entities in a given text using external knowledge bases (Mitchell et al., 2015; Miller, 1995). We have done so based on the following hypothesis:

Hypothesis 2.1. Seeing as we are using the data produced by Hossain et al. (2019), which stems from 2017-18, background knowledge is necessary in order to properly model the humour found in the data set.

2 BACKGROUND

Throughout the thesis we build upon previous work within several areas of research. Most notably is the one of humour classification, but we are building our solutions and answering the research questions by looking at studies addressing different issues too.

2.1 HUMOUR CLASSIFICATION

Due to its complexity and the prerequisite for a deep understanding of the topic, previous research contributions towards automatic humour recognition have been made in selected aspects. The seminal paper by 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. However, the data set produced in the paper, contrary to the one used in this thesis, needs a second document type as the negative examples of humour. This introduces the problem of creating a document classifier rather than a humour classifier. To combat this problem they only selected negative samples that was similar in length to the positive samples, and they tested three different sources of negative examples.

A great resource for finding usable data sets for humour classification is TV-shows such as FRIENDS. Purandare and Litman (2006) capitalised on this and set out to classify each spoken turn of FRIENDS, as either funny or not funny. They defined a funny turn as one where the audience laughs right after the turn. Besides just using lexical features they also analysed the audio from each of the turns, thus also creating acoustic features such as pitch and energy. Using the ADTree algorithm on their lexical, prosody and speaker features, they achieved lower accuracies compared to Mihalcea and Strapparava (2005), however they argue that this is due to their data being homogeneous, i.e. the data being drawn from the same source and their speakers being the same for both classes.

With the rise of social media, comedians and regular people have gotten an outlet for sharing humorous texts, jokes or sarcastic comments. With many types of humour such as wordplay, irony and sarcasm on a platform such as Twitter, the task of categorising each tweet into one of many categories can be quite daunting. Using a semi-supervised learning algorithm Raz (2012) classified tweets into one of 12 predefined categories. They avoid the task of classifying a tweet as either funny or not by sampling their data from a website that share humorous tweets. The unsupervised model found 6 categories of features that it uses to distinguish the 12 predefined categories of humour.

A second place which poses as a great resource for human interaction and a great resource for humorous data is that of Reddit. Weller and Seppi (2019) used data scraped from Reddit to train a model that can assess whether a joke is funny or not. They leveraged the voting system on Reddit to determine which posts was deemed funny and which were not. They, furthermore, demonstrated the effectiveness of the transformer architecture for humour classification.

Humour consists of several latent semantic structures such as Incongruity, Ambiguity and Interpersonal Effect (Yang et al., 2015). Looking into representing these structures is an interesting way to better understand how to discriminate between

humorous and non-humorous texts. Yang et al. (2015) uses a Random forest algorithm to perform humour recognition. They use hand-crafted features build by interpreting the latent semantic structures of humorous texts.

In recent years with the emergence of Deep Learning, the usage of Convolutional and Recurrent Neural Networks for text classification has risen dramatically. Combining Convolutional Neural Networks and Highway Networks, Chen and Soo (2018) focuses on leveraging the power of deep learning for classifying puns, one-liners and short jokes. They sample data from multiple sources, including related works, and from different language sources, such as Chinese, thus creating a very heterogeneous data set for creating robust neural models.

Also using a CNN Chen and Lee (2017) works with a homogeneous data set, which is sampled from TED talks. The task established here is very similar to that of Purandare and Litman (2006), however, here they leverage the CNN to find the features of the utterances rather than creating them, themselves.

Finally, with the extensive work done in Hossain et al. (2019) on humour generation, the goal of their study is to generate a carefully curated data set 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. In their work Hossain et al. (2019) performs an extensive analysis of the different types of humour that are present in the generated data. They gain insights into humour generation strategies (Hossain et al., 2019, see section 3.1) used by their annotators, in fact they find 8 strategies. The strategies includes, creating sarcasm, creating punchlines, creating meaningful n-grams (~~Wall~~ Sesame Street) and creating connections between entities in the headlines (Trump and hair).

The work done in Hossain et al. (2019) has been extended and the data generation has been gamified in Hossain et al. (2020). Here they present a new data set created by crowd sourcing in the form of a game competition. However, contrary to the original data set this has not been carefully curated and is too different to use with the original data set.

2.2 KNOWLEDGE BASES

As part of our humour research we are keen on adding external knowledge to our model for better understanding of the texts. Establishing resources that contains facts regarding the real world is one of the biggest challenges in this endeavour. Knowledge bases such as Freebase[1] and NELL (Never-Ending Language Learning) (Mitchell et al., 2015) are not easy to maintain and they are in constant development to keep up with new facts.

To represent knowledge data in a dense representation, it is needed to develop an intermediate model that can learn such representations. This is very similar to the Word2Vec (Mikolov et al., 2013) models. In the case of knowledge, it is often represented as triplets (*subject, relation, object*) and thus you should create a model that can associate a subject and an object. Liu et al. (2016) developed a Neural Association Model (NAM) that uses the entire triplet and their subsequent representations to predict whether or not it was a true triplet. This creates representations of true relations between objects.

Ahn et al. (2016) creates a Neural Knowledge Language Model (NKLM) which is an extension of a LSTM model. They structure their data into a set of Topic Knowledge, which in turn is sets of facts concerning each topic. At each time step the model then samples a word from either its vocabulary or the Topic Knowl-

[1] Has been succeeded by Wikidata

edge, depending on the context. They furthermore released a data set (*WikiFacts*) of Wikipedia texts aligned with Wikidata[2] facts.

Where Ahn et al. (2016) uses Knowledge Base (KB) to extend and improve a Language model, Miller et al. (2016) uses KBs to improve Question Answering (QA) models. They generalise the Memory Network (Sukhbaatar et al., 2015) architecture by creating a key-value memory structure. The model will then store important knowledge in the key-value memory, which it can then subsequently use for answering given questions.

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 Never-Ending Language Learning (NELL) (Mitchell et al., 2015) and WordNet (Miller, 1995) (a lexical database), in order to integrate the background knowledge and enhance the learning on Bidirectional-LSTM. At each time step the model merges its hidden state with candidate concepts from the external knowledge bases, thus injecting more information than what is available just in the text. They use their implementation for entity and event extraction.

[2] Wikidata: https://www.wikidata.org

3 DATA AND MATERIAL

The data set (Humicroedit) 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 were 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. As reported by Hossain et al. (2019), the scores are correlated with both the headline length - measured as number of tokens present in it - and the relative position of the replaced word within the headline. The humour increased if the edit would happen toward the end of the headline, as per the reconstruction in fig. 1. In order to eliminate the noise, the relative positions have been grouped in 10 bins, as showed in the right-hand side of fig. 1. This approach allows us to spot the increasing trend.

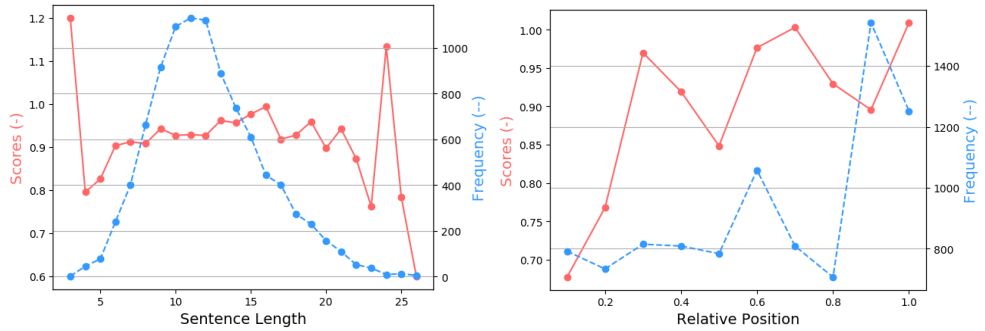

Figure 1: Distributions of headline length and edit relative position

3.1 DATA SET

News headlines, in the humour data set generated in the work of Hossain et al. (2019), were collected from posts on the social media site, Reddit, specifically on two sub-reddits: `r/worldnews` and `r/politics` from January 2017 to May 2018. Only headlines that had a word count between 4-20 words from the top 25 English news sources were kept. The goal of the data collection strategy was to obtain headlines of mainstream and diverse interests, which would presumably make the editing of headlines for humour less difficult. For data annotation process, the U.S. based Mechanical Turk workers were employed to fulfil this purpose. The edited headlines were graded by a group of Mechanical Turks instructed as judges to grade objectively on the edited headline as funny if they believed it would be funny to a large audience regardless of whether it was fun by itself or in relation to the original. Similarly, another group of Mechanical Turks were instructed as editors to edit the original headlines in such a way that the general audience would find them funny without the use of cheap humour (profanity, slang, potty humour and crude sexual references). For each headline, there would be 3 editors assigned, and for each edited headline, 5 judges would be assigned to grade it. Interestingly, it was observed by the researchers that the quality of the edits were declining as they obtain more annotated data (repeated, identical edits), and respectively, some judges were observed to repeatedly assign very low funniness score compared to the 4 other judges. Measures such as randomly sampling a set of judges and editors for each batch, by swapping exhausted Turks out with newer ones, spelling correction

and elimination of concatenation of multiple tokens were deployed to mitigate the aforementioned issues.

In addition to the humour data set (Humicroedit), we also have access to the FunLines data set (Hossain et al., 2020), which utilised game competition as motivation for data generation[1]. Due to some key differences between these two data sets we decided not to include the FunLines data set. The first difference was how the tasks (editing and grading) were not outsourced and performed by a similar group of people, but instead by random participants gathered through the internet, which in comparison to the Turks, appeared less reliable, thus diminishing in resulted data quality. Although the participants were encouraged in both approaches to disregard own bias in the process, one cannot entirely discard them. Vastly different group of participants also introduced a gap of background knowledge, which would be crucial in understanding and generating humour. Another difference that appeared in the FunLines data set upon further inspection was misspelling and repeated counts of concatenation of multiple tokens.

3.2 NELL AND WORDNET

NELL was first developed by Mitchell et al. (2015) to learn basic fundamental semantic relations between entities of many categories. It has since continued to evolve by reading and extracting new instances of entities and relations from hundreds of millions of web pages. The NELL database contains several columns of information, but the columns of our interest are within the triplet: (entity, relation, value). Entity and value embody the concepts or identities that are linked by some relations, which we thought to be relevant in our attempt to integrate background knowledge into the proposed model, as background knowledge about topics, such as events, known relations and entities, were designed to play a significant role in humour generation strategies (see Mitchell et al., 2015, section 3.1)

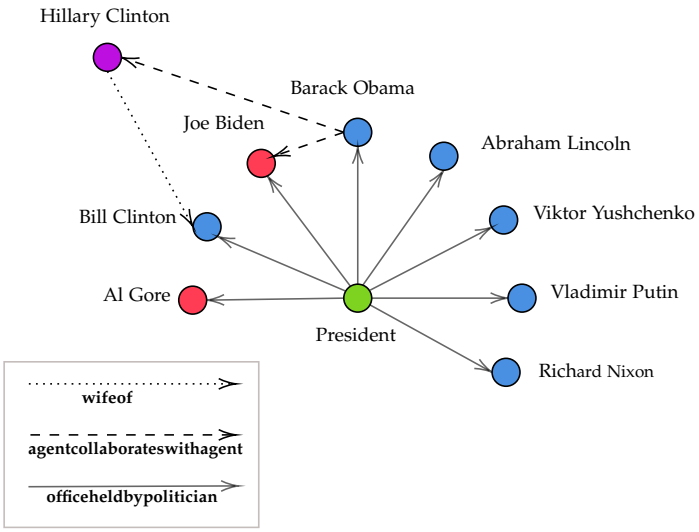

Figure 2: Example of NELL network representation.

As seen in fig. 2, the network snippet shows *Barack Obama*, an entity, is connected to other entities, such as *Hillary Clinton* and *Joe Biden*, by the relation, *agentcollaborateswithagent*. Moreover, *Barack Obama* is also connected to the concept of president by *officeheldbypolitician*, which is also a relation, along with other politicians. However, not every existing connection conveys correct information. Albeit *Joe Biden* and

[1] Funlines: https://funlines.co/humor/

Al Gore's (marked as red) respective appearance as president according to NELL, the highest office held by them so far has been vice president.

As for WordNet (Miller, 1995), it is a large lexical database of English that also contains groups of cognitive synonyms called synsets, which are interlinked by means of conceptual-semantic and lexical relations. Consequently, the network provides additional meaningfully related words and concepts in supplement to those in NELL.

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

Table 1: Entity coverage for training data

For further investigation of NELL, we have conducted analyses to identify potential shortcoming in the database itself and our utilisation of it. First of all, we have investigated how much coverage NELL provides for each headline, in the training subset, as it is, in this case, a good metric for measuring NELL's efficacy. As shown in table 1, the average, median and maximum amount of words NELL is able to cover are respectively 4.8, 5 and 13, which is quite low, even when considering the maximum limit of 20 words as well as NELL's supposedly only regard for entities (nouns). It is also apparent by the minimum amount of zero that some headlines are not covered at all. With the addition of WordNet, we observe a significant increase of 2.4 words on average, a stark increase on the maximum amount covered by 5 as well as a small increase of 1 on the minimum.

Events	Availability
Brexit/ EU referendum (2016-2020)	✗
Meghan Markle (2017)	✗
Trump Campaign (2016)	✗
Donald Trump as politician / president (2016)	✗
Barack Obama as former president (2016)	✗
MS Surface Pro 3 Launch (2014)	✓
FIFA World Cup Brazil (2014)	✓

Table 2: Event list for NELL time scope investigation

Secondly, there is a suspicion that the time scope of the NELL network, belonging to the iteration used in this project, does not seem to align with the time period of the humour data set mentioned in section 3.1. We perceive this as an issue, since relations and entities are dynamic over time as well as the existence of some entities entirely. We have, therefore, investigated NELL with curated events, which took place in 2016 and prior, in table 2. Our findings suggest that the content of NELL might originate from before 2016, as events from 2016 and later do not exist in NELL, whereas both events from 2014 do appear. Although it could, at the same time, also suggest a possibility of an incomplete network.

Token	Part of speech	NELL meaning
as	Adverb / conjunction	Arsenic
a	Article	Biotech company
it	Pronoun	Organisation
he	Pronoun	Helium
will	Verb	Faculty
i	Pronoun	Iodine
him	Pronoun	Music artist
can	Verb	Beverage container

Table 3: Examples of errors by NELL

Thirdly, we have found examples of common error in how NELL interpreted some words for the encoding. In table 3, we have listed the most common errors, which differ significantly from its original meaning in the headlines, along with the supposed Part of Speech (POS) tag and the perceived meaning from NELL.

4 METHOD

In this section we outline the structure of our system and go into details on the different components. We outline a model with two different settings, a base model (fig. 3) and an extended model (fig. 4). The two settings consists of the same three encoders which handles 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. The extended model add a fourth encoder that takes care of the context of the headlines. The fourth encoder is presented in section 4.4. Section 4.5 outlines how the results from each of the three/four encoders are combined and processed. Section 4.6 goes into the training parameters and extra tricks we used to get more performance from our model. To have a reference point to match our model against, we create multiple baselines, all of which is available in section 4.7. Lastly we perform hyper-parameter tuning, which is explained in depth in section 4.8.

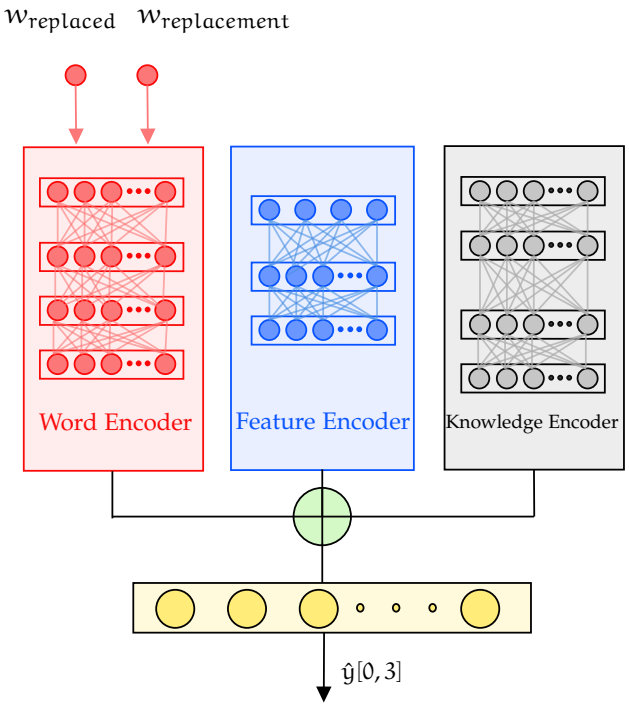

Figure 3: Base model architecture

4.1 WORD ENCODER

The word encoder is handling 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). The processed representations for each of the two words are concatenated before moving on in the model.

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

Table 4: Example of phonetic distance feature showing transcription from grapheme to phoneme.

4.2 FEATURE ENCODER

The feature encoder takes four features that encode humour specific information from the headlines. Each feature helps the model 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).

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. This is consistent with one of the humour generation strategies found by Hossain et al. (2019), which is setup and punchline.

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 perfect 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 4. 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. Thus our hypothesis is that this feature should reflect that.

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 set (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) we can find if the word creates an incongruity.

4.3 KNOWLEDGE ENCODER

The knowledge encoder is searching the headline for any known entities occurring in the NELL (Never Ending Language Learning) database or hypernyms in WordNet. Table 5 lists some example headlines that contain entities, such as named entities (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 with NELL being a

large network, it alone is insufficient in covering a significant part of each headline. In order to expand the coverage, nouns, excluding pre-existing entities in NELL, are extracted from WordNet. 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 improved significantly (table 1).

Each entity is converted to an embedded representation that has been pre-trained using a Neural Association Model (NAM) presented by Liu et al. (2016). The unknown words in the headlines are represented by a zero vector. Finally, entity embeddings get processed by a CBOW model which sums the vectors before they are processed in a FFNN (See appendix A).

The Knowledge encoder is made with the focus of including background knowledge. However, it should further more aid in figuring out if the replacement word creates a strong connection with one of the entities in the headline (Trump and hair)(section 3.1 Hossain et al., 2019, strategy 3).

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

Table 5: Example of headlines from the training data that wouldn't have turned out as fun without the necessary background knowledge. Strike-through (red) denotes a replaced word, italic (blue) denotes a named entity that would benefit from the integration of a knowledge base, like NELL, and green denotes replacement word.

4.4 CONTEXT ENCODER

In the base model configuration only the word to be replaced and the replacement word is used. 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.

Using the Albert model we created an extra experiment where we wanted the Albert model to replace the NNLM in the word encoder. For the experiment we fed the entire headline to the Albert model, and then extracted the replaced words representation from the contextualised word representations produced by Albert. We did the same for the replacement word. Now instead of using the NNLM to encode each of the words we simply input the predefined vectors, thus indirectly encoding the context of the word. The results of this experiment can be seen in table 6.

4.5 OUTPUT

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

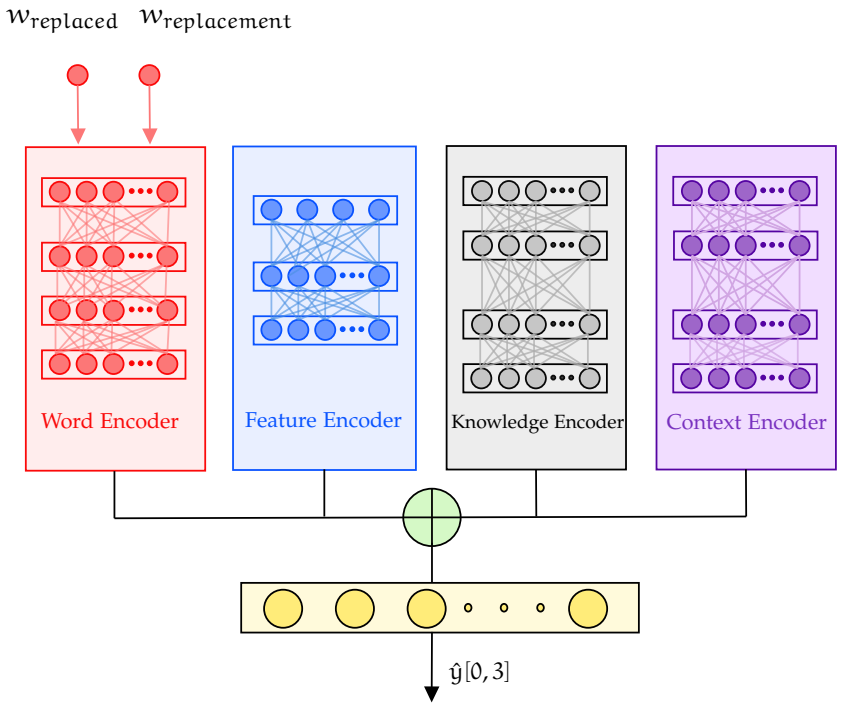

Figure 4: Extended Model architecture with context encoder included

4.6 EXPERIMENTAL SETUP

LIBRARIES We use Keras (Chollet et al., 2015) with the TensorFlow backend (Abadi et al., 2015). The pre-trained models, NNLM[1] and Albert[2], have been 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 full source code for the thesis is available at ITU GitHub[3]. All data needed to reproduce the model is available in the repository.

TRAINING SETUP The Adam optimiser (Kingma and Ba, 2014) is used to optimize the models. The model was trained for 25 epochs where it converges, however we applied early stopping where applicable with patience of 5 epochs. As main evaluation metric we use Root Mean Square Error (RMSE), which for some prediction vector $\hat{y}$ and a true value vector y is:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{1}$$

HYPERPARAMETER TUNING For the subsequent hyperparameter tuning we used the newly created Keras-Tuner[4] library, which is built specifically for Keras. In

[1] NNLM: https://tfhub.dev/google/nnlm-en-dim128/2
[2] Albert: https://tfhub.dev/tensorflow/albert_en_base/1
[3] Source code: https://github.itu.dk/SentimentGroup/HumorHeadlines
[4] Keras-tuner: https://github.com/keras-team/keras-tuner

this library we made use of the Hyperband optimisation strategy explained in section 4.8.

4.7 BASELINES

To make sure that the results we get are due to the features we have created we establish multiple baselines to test our model against.

MEAN/MEDIAN BASELINE The first and most simple baseline is the mean and median score of the training set. Here we compute the mean and median score on the training set and predicts those for each data point.

LINEAR REGRESSION The second baseline we create is a linear regression model that is fitted to our handcrafted features. This is a good baseline to tell us whether the complexity of our model is worth it or we can get by with a simple regression model.

CNN W. HIGHWAY NETWORK To test our model against literature in the area of research we established a similar model to the one presented in Chen and Soo (2018). In Chen and Soo (2018) they use their CNN model for binary humor classification contrary to this study where we are predicting a continuous score. This is a CNN model with 3 convolutional layers and a max pooling layer which is follow by a Highway Network (Srivastava et al., 2015) that consists of 3 Highway layers and a single linear regression output. The filter sizes on the convolutional layers are $(5, 6, 7)$ and the number of filters is 100.

MULTI CHANNEL CNN As a second approach to using CNNs we have created a multi channel CNN model (Kim, 2014). This model uses 4 different embeddings, namely GloVe[5], FastText[6], Google News[7] and Custom Word2Vec trained on our own data. Each of the four embedding types is processed in its own CNN, before the results gets merged together and processed using fully-connected layers.

KBLSTM Seeing as our Knowledge encoder is drawing inspiration from Yang and Mitchell (2017) it is natural to use an implementation of the KBLSTM as a baseline. The implementation used here is taken from this repo[8] and it is a conversion from the original implementation in theano to keras. In Yang and Mitchell (2017) they extend the LSTM architecture to incorporate a knowledge module that at each time step t retrieves candidate concepts from the knowledge base. Using these candidate concepts it creates a new state vector that integrates the candidate concepts and the original state vector from the LSTM. Yang and Mitchell (2017) tests their model on entity extraction and on event extraction, in which it outperforms the related other models.

NNLM In our proposed model we use the NNLM as a word encoder to create encodings for the replaced and replacement word. The NNLM is originally developed as a sentence encoder and thus as a baseline we are able to use it as the encoder and try to regress over the output of it. The NNLM model is followed by 3 fully connected layers with sizes $(64, 32, 16)$ and a single linear regression output layer. The encoding size of the NNLM is 128.

[5] GloVe: https://nlp.stanford.edu/projects/glove/
[6] FastText: https://fasttext.cc/docs/en/english-vectors.html
[7] Google News: https://code.google.com/archive/p/word2vec/
[8] KBLSTM: https://github.com/aianurag09/KBLSTM

4.8 HYPER PARAMETER TUNING

An integral part of developing new machine learning models is making sure to optimise all the parameters to achieve the best possible performance. When we had settled on the structure of our model we set out to perform hyper-parameter tuning. We chose to use the Hyperband algorithm for optimising our parameters. Hyperband (Li et al., 2018) is a Bandit-based approach to the hyper parameter tuning problem. The algorithm extends the SuccessiveHalving algorithm by using it as a subroutine. It does so to automatically choose the number of configurations to try given a finite budget.

The resulting model can be seen in appendix B. The tuning was run over all parameters in the network, and ran for 8 Hyperband iterations. We tested multiple layers in each of the encoders, each layer size, the amount of dropout and the activation function. It was found before hand that adding extra layers to the output did not result in any increase in performance, thus the output was kept to a linear regression style. The resulting score can be seen in table 6. When evaluating the set of parameters we used the development set, thus only predicting on the test set once when we had chosen the optimised parameters. Both the base model and the extended model have been hyper parameter optimised. When we optimised the extended model we fixed the parameters found when optimising the base model as to limit the search space.

5 RESULTS

In this section we present the results gathered from experiments with the base model and the extended model. We test both the base model and the extended model against the seven different baselines outlined in section 4.7. We have trained all models 10 times (where applicable), and the results outlined in table 6 is the mean of those runs and their according standard deviation on the test set.

System	Test Score	Stddev
Mean Baseline	0.57471	(N/A)
Median Baseline	0.59140	(N/A)
Linear Regression	0.57361	(N/A)
CNN W. Highway Network	0.57543	0.00064
Multi-Channel CNN	0.59580	0.00510
KBLSTM	0.57304	0.00119
NNLM	0.56211	0.00111
Base model	0.55510	0.00168
Base model rounded[1]	0.55440	0.00200
Base model Albert Word Encoder[2]	0.57397	0.00066
HP Tuned base model	0.54476	0.00214
Extended model	0.54576	0.00145
HP Tuned extended model	0.54441	0.00214

Table 6: Average scores on the test set. Computed over 10 runs. 1. Experiment where we rounded the predictions to the nearest fifth. 2. Experiment where we used contextualised embeddings produced by Albert as input to the word encoder.

The base model without hyper parameter tuning achieves an RMSE of 0.55510 on the test set. After hyper parameter tuning the base model perform slightly better with an RMSE of 0.54476. By including context (the extended model) we get a barely noticeable increase in performance compared to the not optimised base model reaching an RMSE of 0.54576. Again after hyper parameter tuning we achieve slightly better results with an RMSE of 0.54441.

Because of how the data set is constructed, the possible scores can assume any discrete value from a minimum of 0 to a maximum of 3 with an interval of 0.2. As our model outputs a continuous score, we tested how rounding the predicted value to the closest real possible value performed (i.e. predicted score of 1.37 was rounded to 1.40, and 0.72 to 0.80). The experiment did deliver a very slight improvement. As a matter of fact, we saw a RMSE of 0.55440 compared to the 0.55510 achieved without rounding.

As explained in section 4.4 we made an experiment where we used contextualised embeddings produced by the Albert model. As can be seen in table 6 the model performs significantly worse when using these embeddings instead of the NNLM. The performance is on par with the linear regression model and is worse than the NNLM in itself.

Five of the variations of the developed model outperforms all of our baselines. We can see that three of the neural baselines performs on par with the linear regression and mean/median baselines. Moreover, our median baseline performs worse compared to the mean baseline. However, this is most likely due to the fact that the median is 0.8 and the mean is 0.935, thus the median is farther away from the higher scoring headlines than the mean is. It is too interesting that the multi-channel CNN performs close to the median rather than the mean and linear regression.

6 DISCUSSION

In this section we analyse our findings and attempt to reach a conclusion about the usefulness and shortcomings of the proposed architecture.

6.1 ABLATION STUDIES

In order to help test hypotheses 1.1 and 2.1 an ablation study was conducted. Figure 5 shows the performance on the training and development sets for each of the ablated features of the model without context. For a more detailed overview see appendix C.

An ablation study removes one part of the model or one feature at a time in order to give an indication of how much that part contributes to the performance. If a part of the model is contributing a lot we would expect the performance to get worse if it is removed. If two parts of the model are encoding the same information we expect the performance to remain unchanged when either is removed, but to worsen when they are both removed.

Ablation studies provide a good hint at the part-wise performance and that is why we do one here.

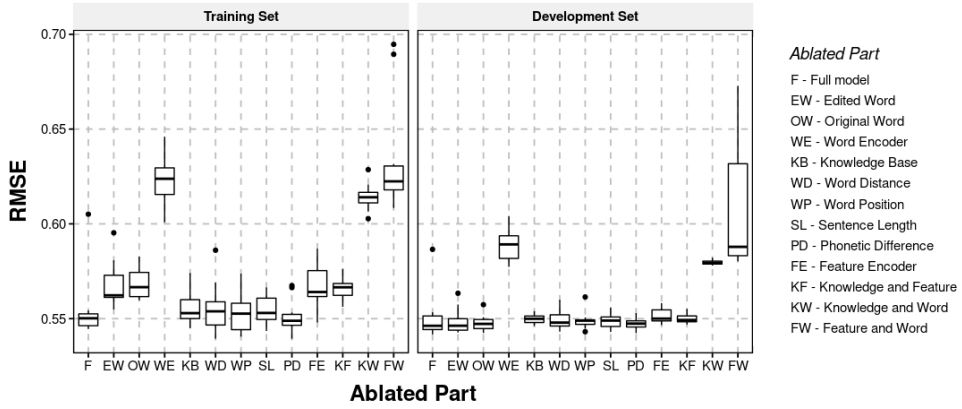

Figure 5: Ablation Study Results. RMSE: Lower is better.

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) but also the most unstable model with the highest variation as the boxplot in Figure 5 reveals. This points at the importance of investigating both mean and median scores in the task.

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

6.2 ANALYSIS OF THE HAND-CRAFTED FEATURES

In order to understand why the apparent impact of the hand-crafted features is so limited we investigate their relation to the true labels.

In fig. 6 we see that there is no discernible pattern between the two. A linear regression line is drawn in blue. It shows that very little information is actually contained in the features, aside from what a mean baseline would contain.

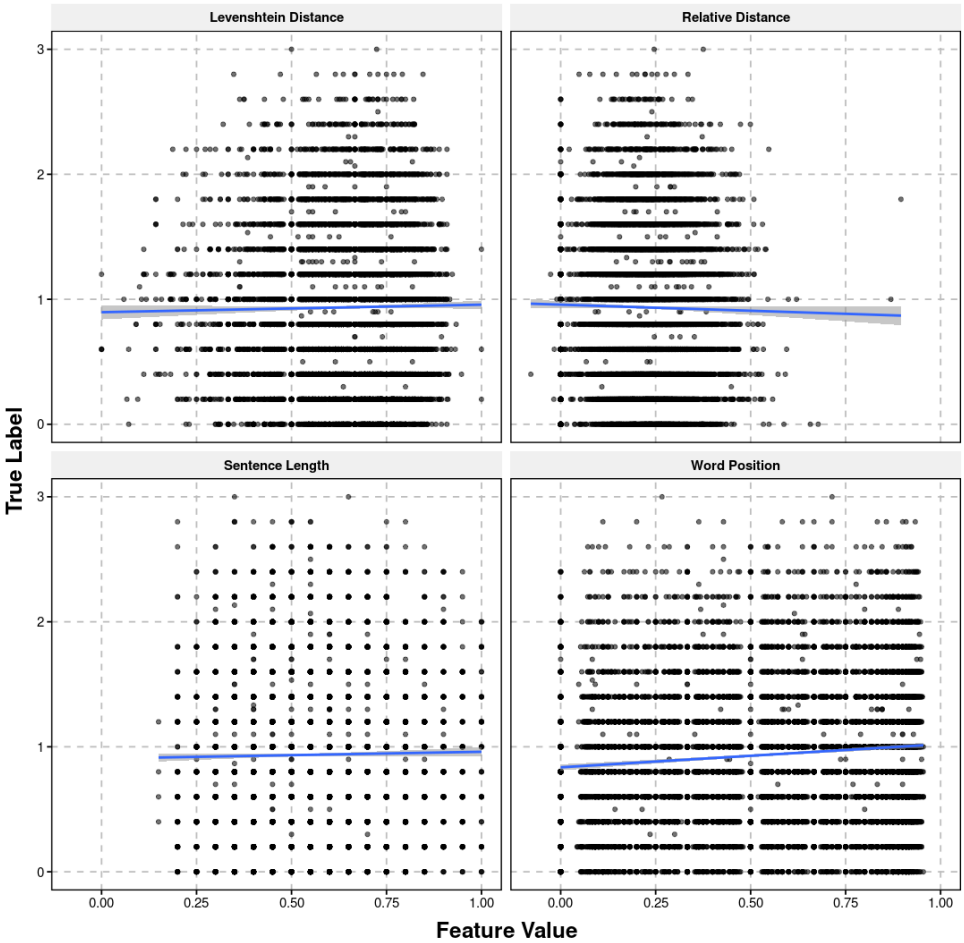

Figure 6: Features against true labels

This goes some way in explaining the results of the ablation study, but it also calls into question some of our underlying assumptions about what makes a sentence funny.

In addition the Sentence Length and Relative Position features look nothing like what we saw in fig. 1. On the features we do not do the pooling described in

chapter 3. This makes for a much noisier output; so noisy in fact, that it appears to not be useful at all.

The relative distance feature also has an issue where, if one or both words are not in the embedding vocabulary they are assigned a distance 0. This is an issue, because distance 0 denotes that the two words are equal. Whereas assigning the words a large distance, would perhaps be more suitable.

This highlights an important limitation of the study. A good future work would be to attempt new features. As well as new ways to implement and integrate the features in the model. In this work we have focused on creating features that fit with the discoveries made in the data set (chapter 3)(Hossain et al., 2019). However, there exists a wide range of related work that goes deeper into the semantic and lexical structure of humour, and from that informs features that better explain humour than ours do.

6.3 PROBING

Following the same line of reasoning presented in the previous section, we investigate why the knowledge encoder does not reflect our assumptions, that a broader factual knowledge is necessary in order to fully understand a joke or a punchline. We analysed the NELL embeddings more in depth, specifically we built a simple classifier aimed at predicting the type of relation existing between two given entities. We took inspiration from the *word content tasks* proposed in Conneau et al. (2018). In order to exclude high and very-low frequency relations, we only analysed 99 out of the existing 831 relations in the NELL database. These are all mid-frequency relations that appear between 700 and 1,300 times in the whole database. This approach resulted in 93,098 records, 65% of which were used for training, 15% for validation and 20% for testing. The two pre-trained entities representation have been concatenated together and used as input in a simple 2-hidden-layers Feed Forward Neural Network that predicts the relation between the entities.

As a result of this analysis we see that the classifier performs relatively well. However, the few major misclassifications we observed are not even remotely similar. For instance, we see class 11 (macro-topic: beverage) and class 61 (automotive business) being misclassifed with class 18 (university), and class 4 (product launching) and 65 (animals) being assigned to class 0 (coaches).

A random guess would have yield an accuracy score of 0.01, while the classifier reached an accuracy of 0.5816 on the training set and 0.5325 on the test set. The latter can be seen in the confusion matrix in Figure 7.

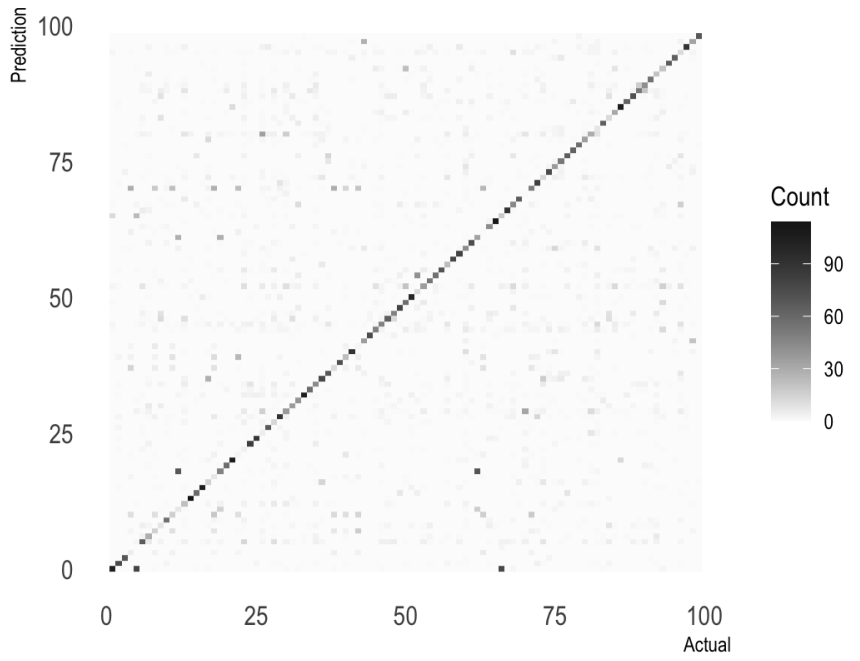

Figure 7: Confusion matrix of relation classifier

We therefore conclude that even though the NELL database carries meaningful information, either those are not actually relevant for the tasks we performed or our knowledge encoder architecture is not able to model it properly.

6.4 KNOWLEDGE-BASE

We set out in RQ 2 to find out whether or not implementing an external knowledge would help the model better understand the content of the text. We hypothesised that we needed background knowledge based on the method of creation of the data set as explained in Hossain et al. (2019)(hypothesis 2.1). However, as shown in our ablation study in section 6.1 our implementation of the knowledge encoder does not seem to impact our performance as much as we had hoped. Our hypothesis for this is that our implementation of the knowledge encoder is not capable of using the data available in our knowledge embeddings (section 6.3).

First we address the weaknesses of our utilisation of NELL: firstly, about how we pick from multiple variations of the same token, and then how we choose from multiple hypernyms. Tokens are generally dealt as single word tokens with the exception of replacement/replaced words, which can consist of two or more words. Consequently, multi-word tokens are misrepresented during encoding. For example, North Korea would instead be considered as two separate words of North and Korea. Another shortcoming lies in the way we match tokens, which only allows exact match during lookup in NELL's vocabulary, as it does not suggest any potential match. Although lemmatisation has mitigated this to a certain degree by grouping together the inflected forms of a word into its lemma, it is far from perfect, as it is does not always lemmatise correctly. Regardless, the shortcoming persists in the form of potential loss for similar words that one might, otherwise, consider a single entity. In the case of multiple variations (in concept or generalisation) of a token, the last variation is always picked, due to the lack of a sophisticated way to accurately

decide the appropriate meaning for the context. Lastly, for tokens with multiple hypernyms, its first hypernym is always chosen to represent the object in a IS-A relation. These shortcomings are a possible consequence of our naive approach in which we utilise NELL and WordNet.

A second inefficiency of our knowledge encoder is the way that we are using the matched knowledge vectors. In the current model we use the knowledge vectors as words in a CBOW model and thus simply sum up the vectors before processing them in a Feed Forward Neural Network (FFNN). We might have been able to get some extra performance if we had used a temporal model like a RNN to process the embeddings. Interesting future work would be to test multiple different ways of applying these knowledge vectors and find which has the highest impact on performance.

6.5 MODEL ANALYSIS

Seeings as we are trying to model a specific distribution, it is interesting too see how well the proposed models and baselines are doing that. Our models all fall quite close to the mean grade baseline, and to analyse why this might be, we have plotted all model predictions as a histogram against the true values histogram in fig. 8. This way we can get a better look on what the models are actually predicting.

As we can see the neural models tend towards predicting the mean baseline score rather than producing a similar distribution similar to the gold labels. Visualising the gold labels in this way show why this task is difficult to solve. Seeing as the score given to a headline is an average of 5 individual scores the resulting distribution of scores will never fill out the entire spectrum but rather have be placed at specific intervals that are a multiple of $1/5$.

In fig. 8 one can see that the *NNLM* and the *MultiCNN* models have potential to model the true distribution seeing as they are the ones of the baselines that have the most spread out predictions.

A solution to this spread out behaviour, and an interesting point of future work, could be changing the type of problem we are looking at. Seeing as each headline is in a discrete class $\{0, 0.2, 0.4, \ldots, 3.0\}$ we could have interpreted it as a classification problem, thus we would have been able to predict some of the outliers.

6.6 CONTEXT ENCODER

In one of the configurations a context encoder was added to the model. It utilises contextual embeddings produced by the Albert model. The performance of this new model architecture is not significantly better than the optimised model. This indicates that the context is either already captured through a different aspect of the model or (more likely) is not modelled in a way that enables the model to use it.

This statement is supported by the ablation study and the feature analysis. It appears that the model is based primarily on the language model, which while it does indirectly encode some context, does not encode any contextual information about the headline. Therefore it seems reasonable to suspect that the context encoder is not abundant. This, in turn, suggest that the context encoder does not in fact encode context in a useful manner to the task.

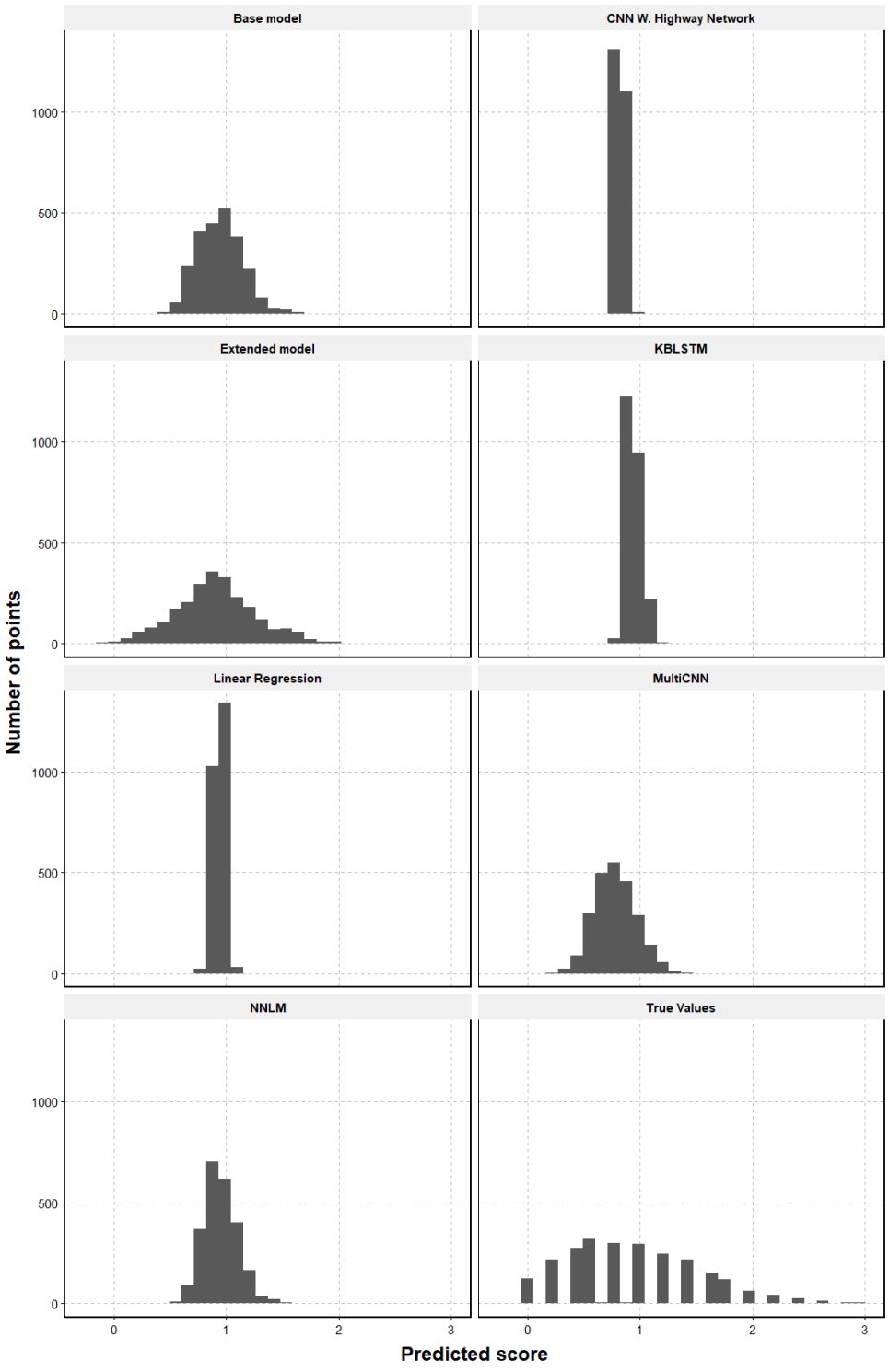

Figure 8: Histograms of predicted values from baseline models

7 CONCLUSION

In this thesis we have developed a model that is able to outperform a variety of baselines. We have been unable to show much effect from the proposed hand-crafted features. In addition our research suggests that they seem to encode very little (if any) information useful to the task. This means we cannot confirm RQ 1. We cannot however determine that it is entirely impossible to find effective features and thus this remains an unresolved issue.

In regards to including external knowledge we have only shown a very limited effect from this. Our exploration of this suggests that the issue is likely either within the underlying data set or in our encoder implementation. We have shown good success in encoding the information contained within the knowledge base in our probing study. But this performance does not appear to manifest in the model, thus suggesting that it is either not compatible with the task or that the implementation in the model is insufficient. Our surface level study shows that there appears to be some shortcomings in the knowledge bases that raises some questions about to which degree they are correct and up-to-date. This leads us to inconclusion in regards to RQ 2. We can show that the effect is limited, but are unable to show whether this is due to knowledge base/task incompatibility, the model implementation being inadequate or due to hypothesis 2.1 being wrong.

BIBLIOGRAPHY

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ahn, S., H. Choi, T. Pärnamaa, and Y. Bengio (2016). A neural knowledge language model. arXiv preprint arXiv:1608.00318.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2003). A neural probabilistic language model. Journal of machine learning research 3(Feb), 1137–1155.

Chen, L. and C. M. Lee (2017). Predicting audience's laughter using convolutional neural network. arXiv preprint arXiv:1702.02584.

Chen, P.-Y. and V.-W. 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), pp. 113–117.

Chollet, F. et al. (2015). Keras. https://keras.io.

Chopra, S., R. Hadsell, and Y. 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, pp. 539–546. IEEE.

Conneau, A., G. Kruszewski, G. Lample, L. Barrault, and M. Baroni (2018). What you can cram into a single vector: Probing sentence embeddings for linguistic properties. arXiv preprint arXiv:1805.01070.

Hossain, N., J. Krumm, and M. Gamon (2019, June). "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), Minneapolis, Minnesota, pp. 133–142.

Hossain, N., J. Krumm, M. Gamon, and H. Kautz (2020). Semeval-2020 Task 7: Assessing humor in edited news headlines. In Proceedings of International Workshop on Semantic Evaluation (SemEval-2020), Barcelona, Spain.

Hossain, N., J. Krumm, T. Sajed, and H. Kautz (2020). Stimulating creativity with funlines: A case study of humor generation in headlines.

Jensen, K. N., N. F. Rasmussen, T. Wang, M. Placenti, and B. Plank (2020). Buhscitu at semeval-2020 task 7: Assessing humour in edited news headlines using hand-crafted features and online knowledge bases. Submitted to the 14th International Workshop on Semantic Evaluation.

Kim, Y. (2014, October). Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1746–1751. Association for Computational Linguistics.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lan, Z., M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut (2019). Albert: A lite bert for self-supervised learning of language representations.

Li, L., K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research 18(185), 1–52.

Liu, Q., H. Jiang, A. Evdokimov, Z.-H. Ling, X. Zhu, S. Wei, and Y. Hu (2016). Probabilistic reasoning via deep learning: Neural association models. arXiv preprint arXiv:1603.07704.

Mihalcea, R. and C. 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, pp. 531–538. Association for Computational Linguistics.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space.

Mikolov, T., E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin (2018). Advances in pre-training distributed word representations. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).

Miller, A., A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston (2016). Key-value memory networks for directly reading documents. arXiv preprint arXiv:1606.03126.

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

Mitchell, T., 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).

Park, K. and J. Kim (2019). g2pe. https://github.com/Kyubyong/g2p.

Purandare, A. and D. 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, pp. 208–215.

Raz, Y. (2012). Automatic humor classification on twitter. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop, NAACL HLT '12, USA, pp. 66–70. Association for Computational Linguistics.

Srivastava, R. K., K. Greff, and J. Schmidhuber (2015). Highway networks.

Sukhbaatar, S., a. szlam, J. Weston, and R. Fergus (2015). End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, pp. 2440–2448. Curran Associates, Inc.

Weller, O. and K. 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), pp. 3612–3616.

Yang, B. and T. Mitchell (2017, July). 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), Vancouver, Canada, pp. 1436–1446. Association for Computational Linguistics.

Yang, D., A. Lavie, C. Dyer, and E. Hovy (2015). Humor recognition and humor anchor extraction. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2367–2376.

APPENDIX

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 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 7: 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.

D PROBING FOR NELL DATABASE

Relation Name	Class ID
coachesteam	0
agentcompeteswithagent	1
locationofitemexistence	2
meetingeventtitleatdate	3
launchingproductcompany	4
softwareisprogrammedinprogramminglanguage	5
statecontainscity	6
inverseofvegetablecanbeservedwithgrain	7
bakedgoodservedwithbeverage	8
scenecontainsobject	9
stateorprovinceresidenceofperson	10
beveragemadefrombeverage	11
athleteplayssport	12
losingscoreofsportsgame	13
wifeof	14
dateofpersonbirth	15
inverseofcriminalssuchascriminals	16
statelocatedincountry	17
universityoperatesinlanguage	18
hasofficeincity	19
inverseofmammalinducesemotion	20
architectssuchasarchitects	21
personinacademicfield	22
automobileenginesuchasautomobileengine	23
crustaceanisatypeofarthropod	24
cityhasstreet	25
inverseofweaponmadeincountry	26
sportsgamescore	27
inverseofriveremptiesintoriver	28
televisionshowisbasedonmovie	29
agentstudiesphysiologicalcondition	30
diseasecausesphysiologicalcondition	31
bodypartwithinbodypart	32
countryhascompanyoffice	33
inverseofarchaeasuchasarchaea	34
roadaccidentcasualtiesnumber	35
meetingeventtitlehasmeetingeventtype	36
cityresidenceofperson	37
organizationnamehasacronym	38
musicalartisthadapetanimal	39
languageschoolincity	40
objectpartofobject	41
arteriessuchasarteries	42
beveragecontainsprotein	43
animaleatfood	44
thinghascolor	45
countrylanguage	46
specializationof	47
hospitalincity	48
academicfieldsuchasacademicfield	49
clothingmadefromplant	50

Relation Name	Class ID
isoneoccurrenceof	51
cityleadbyperson	52
inverseofcoachcanspeaklanguage	53
clothingtogowithclothing	54
vegetableisproducedatcountry	55
inverseofsoftwareprogrammedinprogramminglanguage	56
malecanbethesameascomedian	57
languageofcountry	58
stadiumlocatedincity	59
coachesathlete	60
inverseofautomobilemakerchiefexecutiveceo	61
leaguestadiums	62
mammaleatsinsect	63
locationofpersonbirth	64
arthropodcanbeveryirritatingtomammal	65
hotelincity	66
cityofpersonbirth	67
animalsuchasfish	68
ageofperson	69
inverseofanimaleatvegetable	70
thinghasshape	71
coachesinleague	72
ismultipleof	73
epicenterofearthquake	74
statehascapital	75
agentcreatedorganization	76
attackerinbombing	77
inverseofgrainusedtomakecandy	78
inverseofprofessionssuchasprofessions	79
cityofpersondeath	80
agentcontrols	81
organizationcreatedbyperson	82
superpartoforganization	83
inverseofarteryarisesfromartery	84
teammate	85
inverseofhotelwasbuiltatfarm	86
inverseofawardtrophytournamentwasawardedonstadium	87
politicalgroupofpoliticianus	88
automobilemakercardealersinstateorprovince	89
jobpositionheldbyperson	90
inverseofmalemovedtostateorprovince	91
directorworkedwithactor	92
venueofproductlaunch	93
stadiumoreventvenuedisclosescompany	94
createdbyagent	95
hassister	96
stadiumhometeam	97
producttypeofproductlaunch	98