TAJJEB at SemEval-2018 Task 2: Traditional Approaches Just Do the Job with Emoji Prediction

Angelo Basile
University of Groningen / The Netherlands
University of Malta / Msida, Malta
angelo.basile.17@um.edu.mt

Kenny W. Lino
University of the Basque Country /
San Sebastián, Spain
University of Malta / Msida, Malta
kenny.lino.17@um.edu.mt

Abstract

Emojis are widely used on social media; thus understanding their meaning is important for both practical purposes (e.g. opinion mining, sentiment detection) and theoretical purposes (e.g. How do different L1 speakers use them?, Do they have specific syntax?). This paper presents a set of models that predict an emoji given a tweet as a part of the SemEval-2018 Task 2: Multilingual Emoji Prediction. We built different models and we found that the test results were very different from the validation results.

1 Introduction

To some extent, Twitter can be considered a huge corpus and researchers exploit it in many different ways to get a proxy for various types of annotations. Purver and Battersby (2012) use distant supervision on tweets to build an emotion detection system; Bollen et al. (2011) show that it is possible to use Twitter data to predict the stock market; Mohammad et al. (2016) build a corpus from tweets for modelling stance.

Social media in general are very popular for building corpora for sentiment-related tasks since the users make a wide use of emojis. Pool and Nissim (2016) show that it is possible to achieve state-of-the-art performance in sentiment classification using only automatically gathered data.

The wide use of emojis on the web calls for systems that can automatically process them. When performing opinion mining tasks, for instance, it can be the case that all we have is just an emoji. For example, a single emoji could be used as a reply to an advertisement of a certain product; thus, being able to get the meaning of that emoji is important. Felbo et al. (2017) shows that given a text, it is possible to successfully automatically suggest the most appropriate emoji that can accompany that text.

In this paper we describe our participation in the SemEval-2018 Task 2: Multilingual Emoji Prediction (Barbieri et al., 2018), a challenge that originates from the work of (Barbieri et al., 2017). The task is structured as follows: given a tweet that originally contained one and only one emoji, predict that emoji; the emoji is removed and given as a training label.

We experimented using three different approaches: first we use a shallow feature representation with two different algorithms (Naïve-Bayes and Support Vector Machine); then we experimented with a dense feature representation (i.e. word embeddings) and a deep neural classifier; lastly, we modeled the problem as a translation problem (i.e. treating English and Spanish as the source language and 'Emoji' as the target language) using a state-of-the-art neural translation system to predict the labels as translated sentences.

To summarize, our main contributions presented in this paper are three machine learning models that predict an emoji given some text. We confirm a fact that is well known in the literature, i.e. that neural models can give good results but hyper-parameter tuning is a hard task and if it is not successful, then a good linear classifier with a bag-of-words representation can easily outperform the neural model. Our best English system was ranked 8th in SemEval Shared Task and our best Spanish system was ranked 4th[1].

2 Data

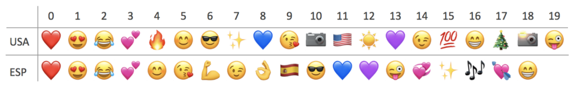
Both the training and testing data are tweet collections: overall, there are 500k English tweets and 100k Spanish tweets, provided as two distinct datasets. These tweets are accompanied by two

[1]We submitted on January 10, 2018. Our submission was made under the name 'The Fabulous EM-LCT Team from Malta'.

sets of labels — one per language — and these labels are the emojis that were originally in the tweet and later removed.

The English label set consists of 20 emojis, while the Spanish label set consists of 19 emojis.

We manually inspected some portions of the datasets and found that the English set contains some Spanish tweets and *vice versa*: this is due to the fact that the tweets were collected automatically using geographical information associated with the account of the user who wrote the tweet. We decided to ignore this fact and assume that the actual test set will also contain some noise. We did not perform any systematic language identification experiment since after the manual inspection it seemed to us that this would not be a problem.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
USA	❤️	😍	😂	💕	🔥	😊	😎	✨	💙	😘	📷	🇺🇸	☀️	💜	😉	💯	😁	🎄	📸	😜
ESP	❤️	😍	😂	💕	😊	😘	💪	😉	👌	🇪🇸	😎	💙	💜	😜	💞	✨	🎶	💘	😁	

Figure 1: The label set for both English (USA) and Spanish (SPA).

The distribution of the labels is highly skewed for both English and Spanish: Figure 2 shows that the most frequent label (i.e. the red heart) is much more frequent than the others.

We decided to conduct experiments with both the original skewed dataset and a balanced version of it. To balance the dataset, we randomly sample from all classes a number of tweets which is equal to the size of the smallest class.

2.1 Normalization

We decided not to perform any sort of normalization on the dataset. The reason for this is that we think that the information that is located at the sub-token level would probably get lost with normalization (e.g. 'amaaaaazing' vs 'amazing'). Character-level differences can be important in helping deciding which emoji is the most appropriate one. For instance, we can imagine that someone who writes 'amaaaaaazing' is more prone to choose a more sophisticated and less common emoji (e.g. a purple heart), while someone who writes in a more sober style would probably pick only the most common emojis.

Furthermore, considering the domain (i.e. Twitter), it is very hard to properly normalize text.

3 Experiments

We performed two groups of experiments: for one group we used a shallow feature representation (i.e. bag-of-words), and for the other we used a dense feature representation (i.e. word embeddings). In the following sections we present these two groups of experiments.

3.1 General Experimental Set-up

In order to facilitate our work, we utilized Python 3.6.3 and set up a Python working environment using `pipenv` to make our experiments easily reproducible; for the same reason we are releasing several Jupyter notebooks containing all the code that we wrote[2]. To train our models, we utilized the standard machine learning package, `scikit-learn` (Pedregosa et al., 2011) for the models using a shallow feature representation. The neural models were built using Keras (Chollet et al., 2015). We used OpenNMT (Klein et al., 2017) to model the problem as a translation problem.

3.2 Traditional BoW Modelling

For our first set of experiments, we chose two different algorithms for text classification and common bag-of-words features.

Algorithms For our algorithms, we worked with Naïve-Bayes and Support Vector Machines. We chose these algorithms in particular because of their success in previous work such as in Wang and Manning (2012).

Features For our features, we worked with the raw counts of each word as well as the tf-idf scores, and n-grams. We used tf-idf to give more weight to the more prominent words within the tweets, and less weight to the more common words, while we used n-grams to capture sequences of words.

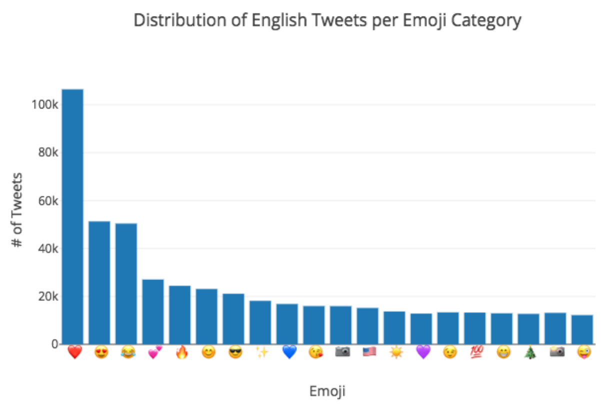
3.2.1 Baseline

We established our baseline by using a bag-of-words approach and the Naïve-Bayes classifier with five-fold cross validation. We chose this as our baseline as we felt that this was the most simple — but still reasonably strong — approach.

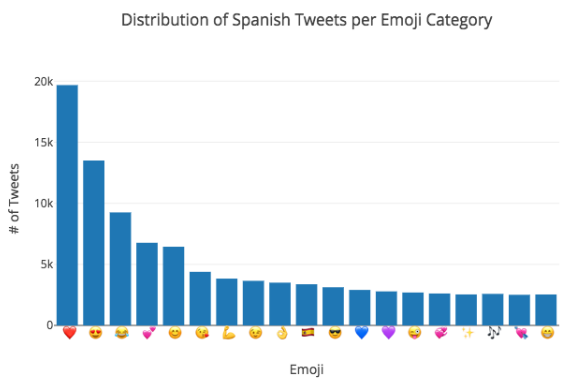
3.2.2 Dimensionality Reduction

As a test, we also looked at the effects of dimensionality reduction using truncated singular value

[2]They can be found here at: https://github.com/anbasile/emojiprediction



(a) English



(b) Spanish

Figure 2: Distribution of emojis in the English and the Spanish datasets.

decomposition (SVD) (also referred to as latent semantic analysis (LSA) when used on textual data) on our best support vector machine model. This was only done with the support vector machine model as it was not possible to do it with the Naïve-Bayes classifier.

We hypothesized that SVD could potentially be useful as it would be able to model topics within the data. If we could model topics within the text, then it would be likely that certain topics would align with certain emojis.

To run the test, we used 10% of our data with dimensions of $n = 100$ and $n = 500$ on our best unbalanced model — the support vector machine model with tf-idf, bi-grams, and POS-tagging. We chose to only use 10% of the data as the runtime of utilizing SVD on the whole English dataset was particularly long.

3.3 Neural Models

In order to take advantage of the large dataset, we decided to try to model the problem using a neural network, which usually requires many instances to be trained properly.

We performed two sets of experiments: first, we trained a simple recurrent neural network. From there, we moved to a more complex recurrent network using bidirectional long short term memory (LSTM) layers in order to account for context from both the right and the left side of words; we then used an already optimized sequence-2-sequence model to model the problem as a translation problem and effectively treating emoji as a language.

The main advantage of using neural networks for processing text is that the feature extraction process and the classification (or structured prediction process) can be optimized at the same time, possibly resulting in better performance.

Embeddings Word embeddings — one of the possible ways to extract feature from text — turned out to be very helpful in NLP and some of its properties (dense, high dimensional format) make it suitable to be used with neural networks. Baroni et al. (2014) have shown that word embeddings almost always perform better when compared to systems using the traditional bag-of-words approach.

To create the embeddings for our experiments, we used the Embedding layer provided in the Keras framework. This method does not make use of the most recent advancements in the field — as those described in Mikolov et al. (2013) — but it has the advantage of being fast and easily understandable.

3.3.1 Predicting Emojis with a Neural Classifier

Using Keras, we set up a network with the structure represented in Figure 3. One way to interpret this kind of modelling is to consider it as a list of matrices. The first layer is the embedding layer, where every node is a word, which is represented as an embedding; the 2 hidden layers are what makes the model able to discover eventual non-linear relationships; the last layer consists of a softmax function that gives in output a vector of probabilities: choosing the most probable class allows to get the most probable emoji.

By making the second hidden layer smaller in length than the external layers, the model implicitly performs a process of dimensionality reduction which allows it to eventually learn latent information (e.g. topic, style, etc.) present in the data.

Furthermore, following the work of Srivastava et al. (2014), we apply to our network a drop-out layer in order to avoid over-fitting and improve the capability of the network to generalize the results.

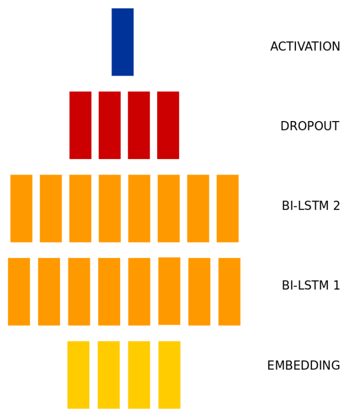


Figure 3: The structure of the neural network.

3.3.2 Seq2Seq Modelling

To some extent, we can say that emojis compress — or tend to compress — all the meaning of the English or Spanish texts that accompanies them. This suggests that the problem can be modelled as a translation problem, where the source language is either English or Spanish and the target language is the emoji language.

We use a state-of-the-art machine translation system and consider our emoji-annotated text as a parallel corpus, having English or Spanish as the source language, and the emojis as the target language.

We use the default network from the OpenNMT framework. This network consists of 2 LSTM layers with 500 hidden units each; furthermore, this implementation includes an attention layer that weights the importance of the different words in translating to the target language[3].

We trained the same network for both English and Spanish — building one model for each language — and we got similar results, even though the English dataset is considerably larger than the Spanish one. Surprisingly, the network learned for both languages that the syntax of the emoji language — so to say — specify that only one token can be used for each sentence.

4 Evaluation

4.1 Metrics

In order to quantify our results for the shallow representation models, we chose to use F-score as our main metric, as F-score is fairly standard and gives us a more unbiased view of the results.

With the dense representation and translation models, we utilized accuracy and perplexity as we could not extract precision and recall for these models for time constraints and technical issues. For the neural networks we did not perform cross-validation since it would have been too costly to perform; however, we use a development set during training and we tested on a blind test set when we finished training.

4.2 Results

Investigating the results of the experiments, we were able to confirm some of our linguistic intuitions.

First, it is evident that the best model for both the English and Spanish dataset based on macro F1 score was the most complex SVM model using tf-idf normalization, part-of-speech tagging and bi-grams. We see that POS-tagging added a substantial increase when comparing the SVM+tfidf model to the SVM + tfidf + 2grams model with an increase of 11% and 5% in the English and Spanish dataset respectively. This suggests that POS-tagging provides key information that helps the classifiers (at least SVMs).

However, dimensionality reduction — with $n = 100$ and $n = 500$ — did not prove to be as successful as we had hypothesized. Instead, we found that it had negative effects on the models as seen in Table 1. This is likely because we remove too much information that distinguishes the individual classes.

Table 1 and Table 2 describe the results for English and Spanish respectively. The models were trained and five-fold cross-validated with various configurations[4]. Models with * were with the bal-

[3]We use the PyTorch implementation: see https://github.com/OpenNMT/OpenNMT-py

[4]The naming of each model is built in the following way: classifier, feature extraction method, n-grams and potential extra features. The following are the acronyms of the configurations– NB: Naive-Bayes; SVM: Support Vector Ma-

anced dataset. The model submitted to the challenge has also been bolded for easy viewing.

Considering the neural models, we found out that the best model using an SVM and bag-of-words outperformed the more sophisticated neural classifier that we built. We believe that this is due to either the hyper-parameters not being optimized. Our networks (emoji-nn-dp) has a structure that is very similar to the one implemented in OpenNMT (except for the attention layer). About OpenNMT: since we got development results that are much higher that test results, we believe to have over-fitted the model. For the OpenNMT network we also report on perplexity since it is a machine translation system and that metric is reported by default. Table 3 highlights the results of the NN. The systems were tested on 100k tweets: we tested our network on the English dataset only because of time constraints. We managed to test the OpenNMT system on both English and Spanish[5].

One of the most exciting things that we observed from our experiments was the impact of the skewed data on the models. In order to see what effect the skewedness had on the data, we tested the best unbalanced model by using a balanced dataset for both languages using the the number of tweets of the least frequent class.

This proved to be a huge success, as observed in the confusion matrix in Figure 4c. As we can see, there is a well-defined diagonal in the confusion matrix as compared to the best unbalanced model in Figure 4b, which shows that the balanced model is more capable of classifying tweets for *all* classes.

If we take a closer look at the matrix, we see that some categories are not retrieved at all. Of particular interest are the heart emojis, represented in categories 0, 3, 8 and 13 for the English dataset. In the best unbalanced model, categories 8 and 13 (blue heart and purple heart) are completely missing while these are actually retrieved by the balanced model. More interestingly, we see that there is a bit of confusion between these four hearts in the balanced model: we think that this is because the distribution of words that co-occur with all of the heart emojis overlap.

chines; tfidf: term-frequency inverse document frequency; POS: part-of-speech tagged data; 2grams: bi-grams; SVD: singular value decomposition.

[5]We are releasing both the OpenNMT systems and an hf5 version of our network with the model and the weights: this will make it easy to reproduce the results without having to train the network again.

In contrast, perhaps the most interesting observation in the balanced model is its success with certain categories. For example, if we take another closer look, we observe that the balanced model performs well at classifying tweets with the fire emoji, American flag, Christmas tree and camera emojis. This is likely due to the fact that these emojis, especially the American flag and the Christmas tree, have very predictable distributions. For example, the tweet 'Things got a little festive at the office #christmas2016 @ RedRock...' may be easy to classify because of the word 'festive' or because of the hashtag.

model	accuracy	precision	recall	macro F1
baseline (NB + bow)	0.30	0.39	0.30	0.21
NB + tfidf + POS + 2grams	0.24	0.57	0.24	0.13
SVM + bow	0.31	0.28	0.31	0.29
SVM + tfidf	0.33	0.30	0.33	0.30
SVM + tfidf + POS	0.43	0.41	0.43	0.41
SVM+ tfidf + 2grams	0.35	0.31	0.35	0.32
SVM+ tfidf + POS + 2grams	0.45	0.43	0.45	0.43
SVM + tfidf + POS + 2grams + SVD	0.31	0.23	0.31	0.23
NB + tfidf + POS + 2grams*	0.33	0.32	0.33	0.32
SVM + tfidf + POS + 2grams*	**0.35**	**0.34**	**0.35**	**0.34**

Table 1: Results of English Models.

model	accuracy	precision	recall	macro F1
baseline (NB + bow)	0.26	0.32	0.26	0.17
NB + tfidf + POS + 2grams	0.22	0.37	0.22	0.10
SVM + bow	0.23	0.21	0.23	0.21
SVM + tfidf	0.25	0.22	0.25	0.23
SVM + tfidf + POS	0.30	0.27	0.30	0.28
SVM + tfidf + 2grams	0.26	0.23	0.26	0.24
SVM + tfidf + POS + 2grams	0.31	0.28	0.31	0.29
NB + tfidf + POS + 2grams*	0.22	0.21	0.22	0.20
SVM + tfidf + POS + 2grams*	**0.21**	**0.20**	**0.21**	**0.21**

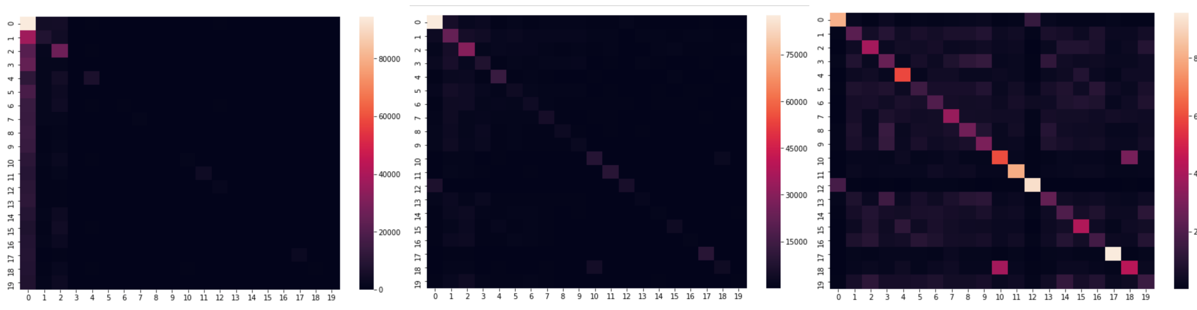
Table 2: Results of Spanish Models.

model	accuracy	perplexity
emoji-nn	38.71	-
emoji-nn-dp	34.23	-
OpenNMT-en	65.11	3.29
OpenNMT-es	71.02	3.28

Table 3: The Result of Neural Network Models.

4.3 Test Results

All the previous discussion about the results is based solely on cross-validation (for the NB and SVM based models) and validation data (for the neural models). The results based on the test set show that we ended up over-fitting the neural models and that the high-scores obtained during validation do not hold for the test set: the model that looked like to be the best during validation (i.e.



(a) Baseline (b) Best unbalanced model (c) Balanced model

Figure 4: Confusion matrices of select English models.

OpenNMT-es, 71.02% accuracy) saw its performance dropping by over 50%. From investigation of the test set results, it seems that the loss in performance may also be due to the fact that the translation model is conflating the different variation of hearts with the more ‘plain’ red heart, as we observe this in the Spanish results.

5 Conclusions

To summarize, we performed several machine learning experiments on an automatically created corpus: we experimented with both traditional bow models and with recent neural models. We conclude that when optimized properly, the neural models can outperform linear classifiers; also, we have to note that training neural models requires a significant amount of computational power, which is not always available. When a big amount of data is available, neural networks usually tend to perform better than other models.

Due to time constraints, we did not perform a systematic optimization of the hyper-parameters of the models and instead we only tried some options that are know in the literature to work well (e.g. using a linear kernel for the SVM, since text data are usually linearly separable). Furthermore, we did not try using pre-trained embeddings with neither models: in the future we plan to use the Glove (Pennington et al., 2014) embeddings trained on Twitter. In particular, the next major experiment that we will try will be the following: modelling the problem as an image description problem. Considering the nature of the labels — emoji — it is easy to see that they cannot be treated as discrete indistinct labels, but instead they share many features that are easy to represent visually: as an example, we can clearly imagine that a camera with flash and no flash or a purple and a red heart are much more similar than, for instance, a red heart and a smile. We had this idea during the last stage of conducting our experiments and for time reasons again we could not test this idea.

Acknowledgments

We are thankful to Amazon for providing us the AWS Educate Grant that allowed us to conduct all our experiments. Furthermore, this paper originates from an assignment for an exam at the University of Malta, where we are both enrolled as students. We are thankful to our supervisors J.P. Ebejer and Claudia Borg for their suggestions and feedback. We also would like to thank our reviewers for their detailed suggestions.

As an aside, we really wanted to find a name for our system that would link our experiences in Malta to this project and landed on *tajjeb*. *Tajjeb* is Maltese for ‘good’, which we think accurately depicts our work here. If you’ve notice, ‘prediction’ begins with a ‘P’ and not a ‘B’, but we chose it as it was the most appropriate word for the acronym and because Maltese has word-final obstruent devoicing (thus reflecting how it actually sounds!)

References

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 105–111. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, United States. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247.

Johan Bollen, Alberto Pepe, and Huina Mao. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *CoRR*, abs/0911.1583.

François Chollet et al. 2015. Keras. `https://github.com/keras-team/keras`.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. 2016. A dataset for detecting stance in tweets. In *LREC*.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Chris Pool and Malvina Nissim. 2016. Distant supervision for emotion detection using facebook reactions. *CoRR*, abs/1611.02988.

Matthew Purver and Stuart Adam Battersby. 2012. Experimenting with distant supervision for emotion classification. In *EACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.