COMP5046 Assignment 2

Max Byrnes and Zhuoxi Kuang (Group 90)

University of Sydney, NSW 2006, Australia
{mbyr2845,zkua9391}@uni.sydney.edu.au

1 Data preprocessing

In order to create a slot tagging model for in-game dota chat, a series of preprocessing methods were implemented on the data. Text in all the training, test and validation datasets were converted to lowercase. This is because for our implementation of word embeddings, it is necessary to unify the same words independent of case. For example, without lowercase conversion, the initialism 'WTF' would have a different vector representation to 'wtf' despite being semantically and syntactically equivalent. As each sentence was also represented in terms of the conversion from words to tags in the training and validation datasets, the sentence and labels were split at each instance of a whitespace character. This essentially tokenized the sentences and labels by converting them into a list. Furthermore, the data in the train, validation and test sets also experienced length reduction where applicable. This process involves deleting additional characters for each word if the total count of a character sequence surpasses 3. Data from in-game chat can be very messy, so this approach attempts to standardize the spelling of words that would refer to the same concept but represented differently due to emphasis with arbitrary character repetition. For example, with this preprocessing technique the word 'wowwwwww' would be converted to 'wowww'. Exploring the 'messiness' of in-game chat further, misspellings of words are very frequent as players can typically interpret what a misspelt word was actually referring to and thus accuracy is not important. For a text classification model, this is generally not the case. As a method of aiding our text classification model, a unique word list generated from the training and validation datasets was created. The words present in the test set were subsequently checked against this list. If a word from the test set did not appear in the list, then the word list would again be scanned, but for similarly spelt words (defined as words with an edit-distance of 1) [1]. If a similar word was present in the word list, then that word would replace the oov word in the test set. Because unknown words of short length are likely to be within one edit-distance of a whole range of words, this preprocessing method only occurred for words of length greater than three. For example, if the word 'wizard' appeared in the train-validation word list, and there was an oov word in the test set, 'wizrd', then this test-set word would be converted to 'wizard' as the words have an edit distance of 1 (add 'a'). This edit-distance method is not effective for short words, as a word such as 'waw' has many common words which are similar ('wow', 'war', 'law', 'raw'). Although the removal of punctuation and numbers are common in many nlp tasks, in this case there were 'words' which were purely punctuation or numbers, so it was important to preserve this type of text data.

2 Input Embedding

In order for text to be processed as input in the models, they must first given a vector representation. This includes semantic, syntactic and domain-specific embeddings. Words can be given semantic embeddings where 'meaning' is represented through association with other sets of words by association. This is based on the idea that words which tend to be grouped together in a similar context tend to have a similar meaning. For this, we processed the training and validation sentences with FastText to generate semantic embeddings. To generate these word embeddings, each word is processed with reference to surrounding words within a specified neighbourhood (window size). Because we were dealing with relatively short sentences, the vector of each word was generated from a window size of just 3. For the creation of our word embeddings, a dimension size of 100 was specified. FastText was used in this case because this breaks down words into its components to allow for the creation of embeddings from OOV words. The advantage of this method over alternatives such as CBOW and SkipGram, is that it is expected that a lot of words in the test set will be OOV due to the nature of the data being raw in-game chat, while these alternatives cannot deal as well with unknown words. Due to further implementation of semantic embedding through domain-specific areas, only the training and validation datasets were used in this FastText model.

For the syntactic embeddings, the grammatical elements of words are intended to be modelled. This can be done through evaluating the position of different words in a sentence. In these cases, it is necessary to model each sentence as opposed to solely singular words because there exists words that can take on different forms depending on context (e.g. 'sign' in 'I saw the sign' and 'I saw him sign it'). The first syntactic embedding was part-of-speech (PoS) tagging. Using a pre-trained spaCy model, the words in each sentence were classed as different parts of speech, such as nouns, verbs, proper nouns, symbols etc. The reasoning behind this is that there would be grammatical patterns with respect to the types of words that are classed as being either toxic, dota characters or dota terminologies. For example, it would be expected that dota characters are nouns as opposed to verbs, and words that are numbers or symbols would not be toxic language. Dependency-parsing was also used in this slot-filling model. The idea behind this is to model the intra-sentence grammatical structure. For example, a word that is a preposition is unlikely to be a dota character, as opposed to a nominal subject. In both cases, the representation of syntactic class was done through the construction of one-hot encoding where each class was represented by a unique vector of equal length with one '1' and other numbers as '0'. This was feasible in this context, because the number of classes was relatively low (e.g. just 17 for PoS tags) so it was not too computationally expensive to attach the set of encodings for a given sentence in our model. In total our dimensions for these syntactic features for our input embeddings was 62.

In order to make our slot-filling model optimal in its performance, it is necessary to further formulate embeddings based on domain-specific documents. For example, using word-embedding model pretrained on more general text such as Wikipedia or

twitter would have less power on this dota chat dataset as opposed to a model strictly trained on information relating to dota and its in-game chat. For this aspect, we scraped strategy guides for every dota character from the dota fandom wiki [2] and applied FastText on this model to generate dota-specific embeddings. Because the text was relatively long for each entry. A wider window size was applied (10), with a dimension of 50. Furthermore, in-game chat was accrued from the CONDA-test dataset as introduced by Weld et al. (2021) [3]. This dataset was subject to a FastText embedding with a window size of 3 (because each sentence was relatively short) with a word dimension of 100. For all of these embeddings discussed, they were concatenated for each word in order to serve as input into the slot-tagging model. In total our input embedding dimension was 312.

3 Slot Filling/Tagging model

The task at hand was to tag each token in the set of test sentences as a specific entity (NER). Therefore, because we are dealing with identical input and tag spaces, our model is a sequence-to-sequence model. A Bi-LSTM model was chosen to process inputs from both directions which enables for the preservation of information from the past and future points of a sentence. Initially, we believed that a model with 2 stacked layers, and a single layer of self-attention calculated with a scaled-dot product (Eq.1) along with a CRF attachment would be a good estimate at the best model. This is in comparison to the baseline model which was made up of only a single stack with a CRF attachment.

Our original hypothesis was that a CRF attachment would be superior to a model without a CRF attachment. This was because a CRF attachment probabilistically selects the optimal tag sequencing of the input as opposed to just merely the resulting tag from the Bi-LSTM output layer for each token. Therefore, if there is some level of uncertainty in which tag is given to a token, this can help to be resolved through determining the most probable tag sequence.

For attention, we looked at it in the context of a two-layered Bi-LSTM model with the CRF attachment. We believed self-attention would improve upon a baseline because it helps to formulate an output token prediction based upon the important points of a sentence, specifically in the context of predicting a sequence that we know the length of (as opposed to tasks such as machine translation where it may be better for an encoder-decoder attention model) as specified in Vaswani et al. 2017 [4]. We also hypothesized that the optimal attention calculation would be the scaled dot product. Because we are dealing with quite a high dimensionality for our word embeddings, scaling the dot product by a factor of the inverse of the square root of the embedding dimension, can alleviate issues with the model's convergence. We later compared the calculations of a scaled-dot-product with a standard dot-product (Eq. 2) and cosine similarity (Eq. 3). The cosine calculation here forms attention scores based on similarities of vector direction as opposed to magnitude.

Two stacked layers was also hypothesized as an optimal model because it was believed that one layer may not be able to accurately model more nuanced relationships between the input vectors. Two stacked layers in the context of a Bi-LSTM model refers to the output of the first hidden layer in the Bi-LSTM for a single direction acting as an input into a second hidden layer from the same direction. For the output, the final hidden layers from both directions are concatenated for each input. The addition of a second layer also opens up more possibilities to place attention mechanisms to model which elements of a sentence are the most important for the classification of a token in different layers. We didn't test over 2 layers, because we felt that based upon the relatively short nature of the input sequences, and narrow semantic context, the addition of more layers may lead to overfitting. For testing the performance of different attention positions within the context of 2 stacked layers, we tried one attention layer after the 1st stack, one attention layer after the 2nd stack, and an attention layer after each stack where both matrices were averaged before calculating the attention score. This was because we assumed that maybe a combination of information from either layer could be helpful for calculating the attention scores. Ultimately, from our analysis we found that the most optimal number of stacked layers was 1. This is perhaps because the relationship between input and output tokens was not as complex as we thought, which seems likely with many sentences that are between 1 and 3 words long.

$$a(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{d_k}} \quad \text{[Eq. 1]}$$

$$a(s_t, h_i) = s_t^T h_i \quad \text{[Eq. 2]}$$

$$a(s_t, h_i) = \cos(s_t, h_i) \quad \text{[Eq. 3]}$$

4 Evaluation

4.1 Evaluation setup

For measuring the performance of our models, we used the micro F1-score for the overall categories as well as the F1-scores for each individual entity. The tokens evaluated involves toxic words (T), in-game slang (S), dota characters (C), dota-specific terms (D), pronouns (P), and Other (O). Additionally, there was a [SEPA] token which was included in the total F1-score but omitted from the token-specific F1-score because the models generally showed a performance of 100% because there was a one-to-one mapping i.e. SEPA was used in the dataset to denote when a player typed a message out over several lines. Ultimately, the evaluation of each model will be looked at by the total performance over the tag set, while comparisons between categories are also used for measuring the effectiveness of the model on specific aspects of the data. The formula described the calculation of this total F1-score is given in Eq.4 where n is a tag

category in the set N, $\widehat{w}_n$ is the number of words predicted as being in n, w_N is the number of all words in the set, and $TF1(n)$ is the F1 score for class n.

$$\boldsymbol{TF1} = \sum_{n \in N} TF1(n) \frac{\widehat{w}_n}{w_N} \qquad \text{[Eq. 4]}$$

We not only solely focus on the performance of our proposed model, but we also alter different components of our model ceteris paribus to determine the specific contributions of each element to our model's performance. This is the ablation study, which consists of altering our embedding model, followed by attention score calculations methods, number of stacked layers, the inclusion of a CRF layer, and the comparison between the best model from this analysis and the basline model. Ultimately, the best model determined by the ablation study (i.e. a model utilizing attention) will be compared to a baseline model (without attention). The hyperparameters set, which remained unchanged for every model included in our analysis, was a hidden dimension of 250, a dropout of 0.5 for each layer, and a weight decay of 0.0001 with the use of the SGD optimiser. Additionally, 2 epochs were used because according to previous research this seemed to be the point in which the performance levelled out . As a result of using only 2 epochs, the learning rate was set at 0.1.

4.2 Evaluation result

As the first subject of our ablation study, we tested three sets of input embedding models. Unchanged over each set of input embeddings included the decision to use FastText on the training and validation data. The first combination included the full range of input embeddings (FastText on the training and validation data, FastText on the dota wiki and CONDA test datasets for a domain-specific embedding, pos tags and dependency parsing). From the results in the table below, this approach resulted in the highest total F1 score (99.33%), along with all specific individual tags apart from game slang. Out of the listed tags in the table, it performed best on the Pronoun tag, while the worst performance was on the dota-specific tag. The performance on the pronoun tag is likely because these tags were very narrow in terms of the domain of associated tokens. For example, words such as 'you', 'u', 'I' are extremely frequent throughout the dataset with little variation so the performance should be relatively higher than the other (non-SEPA) categories. The fact that the dota-specific tag had the worst performance is likely because the domain-specific documents were not comprehensive enough in terms of forming a large contextualized vocabulary, in addition to there being much fewer instances of these tokens in the training data leading to greater difficulty in detecting patterns that emerge with this category. Over the set of all input embedding combinations, the performance on the dota-specific tag was still the highest at 94.36%.
When the PoS tags and dependency parsing was removed from the input embedding, the total F1 score performance slightly dropped to 99.24%. In this instance, the input embedding dimension dropped from 312 to 250. From the table, it can be seen that this input embedding technique performed worse than the full range of input embeddings

over all sets of specified individual tokens. The largest drop was in the toxic word category which saw a drop from 99.07% to just 96.70%. This is likely because the usage of toxic language in game follows a relative strict grammatical format, where OOV toxicity can be inferred from PoS tags and dependency parsing.

The next input embedding combination was very similar to the first combination but lacked the CONDA test dataset for its domain embedding. This resulting dropped the input embedding dimensions from 312 to 212. The performance was the worst out of all input embeddings, dropping to 99.07% for the total set of tags. Interestingly, it still yielded the best performance over the in-game slang tag. This was quite unexpected because the removed CONDA test dataset was of players' in-game chat, in which slang would have been much more frequent as opposed to the dota wiki documents which focused solely on game mechanics such as strategy, characters, abilities and items. There was additionally a significant drop in the performance on modelling the dota-specific tokens (87.23%) as opposed to 93.22% for case of no syntactic embeddings. This shows that although it was expected that the dota wiki documents would aid in the tagging of dota-specific tokens, without the addition of the CONDA test set, the performance still drops significantly. As expected, it can be seen that the optimal input embedding setup is that which features the full-selection of embeddings (FastText on training and validation, PoS tags, dependency parsing and FastText on the dota wiki and CONDA test datasets).

	FastText (training + validation) + pos + dep + FastText (dota wiki + CONDA test)	FastText (training + validation) + FastText (dota wiki + CONDA test)	FastText (training + validation) + pos + dep + FastText (dota wiki)
T-F1	**99.33%**	**99.24%**	**99.07%**
T-F1(T)	99.07%	96.70%	95.17%
T-F1(S)	99.01%	98.92%	99.26%
T-F1(C)	97.39%	97.25%	96.68%
T-F1(D)	94.36%	93.22%	87.23%
T-F1(P)	99.83%	99.78%	99.63%
T-F1(O)	99.59%	99.54%	99.47%

Table 1 – Ablation study of input embeddings

As our proposed model was to include attention, our next stage of the ablation study focused on the attention-specific aspects of the model. This included the attention calculation methods, and the position of attention within our stacked Bi-LSTM model. In this case the parameters included the best embedding model (including part-of-speech tagging and dependency parsing), with the same types of hyperparameters. To test the attention calculation formula, we used attention from after the final layer from a two-stack Bi-LSTM. We found that the optimal performance was achieved with the scaled dot-product as previously hypothesized. We believe this was because we were dealing with high-dimensional data for our embeddings so scaling the dot product would aid in

the convergence of the model. This however, was only a very slight improvement over the standard dot-product calculation. Performance of the cosine similarity calculation was only 99.04% which was by far the worst performing calculation. Therefore, we would include the scaled-dot product in the final model. For testing our attention position, we had three options for a two-stack Bi-LSTM model. This was placing attention after the first layer, after the second
layer, and after both layers at the same time. We believed that placing attention after the second layer would perform the best because the output from the Bi-LSTM model using two-stacked would be most indicative of modelling the relationship between output tags and input tokens as opposed to just using the first hidden layer. Judging from the results, the F1-score for attention in solely the second layer was the highest (99.25%). The next highest was attention in the first layer (99.18%), followed by the inclusion of two attention layers after each hidden layer of the Bi-LSTM which are then averaged (99.01%). We believed that including two attention layers may significantly reduce results primarily because not only are we dealing with short sentences but also that averaging the two attention layers would likely dampen the resulting softmax resulting from the attention in the second layer.

Model	Strategy	F_1 score
Bi-LSTM + Attention + CRF	dot-product	99.21%
	scaled dot-product	99.25%
	Cosine similarity	99.04%
Bi-LSTM + Attention (scaled dot-product) + CRF	Attention in first layer	99.18%
	Attention in second layer	99.25%
	Attention in first and second layer	99.01%

Table 2 – Ablation study of different attention strategies

Now that we identified the optimal attention setup is using a scaled dot-product after the final hidden layer, we attempted to test how our model performed upon changed the number of stacked layers. Interestingly, model performance declined with the introduction of more stacked layers. Although we initially hypothesized that using two stacks would result in an optimal performance, the F1 score (99.33%) was lower than that of using a single stack (99.45%). The decline was much more drastic for the case of using three stacks (98.26%). This is perhaps because we used quite a large hidden dimension (250), so the inclusion of more than one stack would likely have reduced performance due to overfitting. Therefore, our chosen model would be defined by a single-stack Bi-LSTM model with self-attention calculated by a scaled dot product.

Model	Stacked layers	F_1 score
Bi-LSTM + Attention (scaled dot-product) + CRF	1 stacked layer	99.45%
	2 stacked layers	99.33%
	3 stacked layers	98.26%

Table 3 – Ablation study about different stacked layers

From this, our best model has been formulated: a single stack Bi-LSTM model with a layer of self-attention calculated by scaled dot-product, CRF and making use of the full set of input embeddings. In comparison to the baseline model, the only main difference is to do with the inclusion of scaled dot product self-attention in the base model. Although we expected the best model of our tested models to achieve a superior performance in comparison to the baseline model, it performed slightly worse. For the total performance, the baseline achieved an F1 score of 99.54% while the best model only achieved a performance of 99.45%. Although the difference in performance seems very slim, the baseline model performed superiorly over each tag category. In terms of the dota-specific domain of the dataset, the most difficult aspect was initially identified as correctly classifying words as either being toxic, in-game slang, dota characters, or other dota terminology. For example, the category of token in the best model which had the lowest F1 score was that pertaining to miscellaneous dota terminology. For the baseline model, this score increased significantly to 97.92%, an increase of 2.68%. The next highest increase was 0.65% pertaining to the toxic tag, followed by the character tag (0.49%). Game slang (S), pronouns (P) and the 'other' tag (O) achieved an F1 score of over 99% for both models, although there is a slight increase in performance for the baseline model.

Model	**T-F1**	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
Baseline	**99.54%**	97.89%	99.56%	98.52%	97.92%	99.92%	99.64%
Best Model	**99.45%**	97.25%	99.46%	98.03%	95.24%	99.89%	99.61%

Table 4 – Comparison between our best model and the baseline

In this case, the exclusion of attention seemed to result in a higher performance. The reasoning behind this may be because the text data we were dealing with were very short sentences which are typically only a few words long, therefore the use of attention to highlight important parts of sentences may have been more limited in comparison to just using the baseline Bi-LSTM.

From our results, it can be seen that both the 'best' model and the baseline model faced relative difficulty in correctly tagging words that were toxic and dota-specific terms. In order to improve the score further, we believe that it is important to be able to target these specific categories with more data. For example, the domain-specific datasets used were not primarily suited for detection of toxicity, but rather in game

phrases and slang. To improve upon the detection of toxic comments, approaches by ElSherief et al. (2018) [5] on determining toxicity on twitter, and Märtens et al. (2015) [6] in the domain of toxicity detection in online games. With a limited number of items, characters and character abilities in dota, a good approach is to potential create a gazeteer of terms to help classify these dota-specific terms. For example, this could be achieved by using the public dataset on Kaggle [7] which also features an enormous amount of in-game chat data which can further be applied in the creation of a future tagging model.

References

[1] Jurafsky, D., and Martin, J. H. (2022). Speech and Language Processing. (3rd ed.) Unpublished manuscript. https://web.stanford.edu/~jurafsky/slp3/

[2] Dota 2 Wiki., Retrieved May 19, 2022 from https://dota2.fan dom.com/wiki/Dota

[3] Weld, H., Huang, G., Lee, J., Zhang, T., Wang, K., Guo, X., Long, S., Poon, J., Han, S. (2021). CONDA: a CONtextual Dual-Annotated dataset for in-game toxicity understanding and detection. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, (pp. 2406-2416). Association for Computational Linguistics.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uskoreit, J., Jones., L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems, 30*.

[5] ElSherief, M., Nilizadeh, S., Nguyen, D., Vigna, G., and Belding, E. 2018. Peer to peer hate: Hate speech instigators and their targets. *Proceedings of the International AAAI Conference on Web and Social Media, 12*.

[6] Märtens, M., Shen, S., Iosup, A., and Kuipers, F. 2015. Toxicity detection in mult iplayer online games. *2015 International Workshop on Network and Systems Support for Games (NetGames)*, (pp. 1–6). IEEE.

[7] Anzelmo, D. (2019). Dota 2 Matches, Version 1. Retrieved June 1, 2022 from https://www.kaggle.com/datasets/devinanzelmo/dota-2-matches