COMP5046 A2: In-game Toxicity Detection

Brandon Chen[1] and Faye Xie[2]

[1] University of Sydney, NSW 2006, Australia
`tche8310@uni.sydney.edu.au`
[2] University of Sydney, NSW 2006, Australia
`nxie0271@uni.sydney.edu.au`

1 Data Preprocessing

For data preprocessing, we conduct lowercase conversion and tokenisation on all data including training, validating and testing datasets. All input were first processed by applying the lowercase filter as this method is commonly used in text classification evident in [3]. This is useful because words with variants such as *wow*, *Wow*, *WOW*, and *WoW* should all have the same meaning. If we don't convert them into lowercase, then each variant of *wow* will have different word embedding. However, since they should have the same meaning and the same slot tag, they should have the same word embedding vector representation (under Word2Vec scenario).
This is followed by tokenising each word by space, this allowed each word to be differentiated from another. In addition, since In-game Toxicity Detection is a Named Entity Recognition task, for each word in a sentence, it has a corresponding tag, hence, using space as delimiter to tokenise sentence is the best choice. Moreover, start and end tokens for sentences were not required as the models produced are trained by stochastic gradient descent where batch size is 1.

2 Input Embedding

2.1 Aspect 1: Syntactic Textual Feature Embedding

We experimented attaching the part-of-speech (POS) tags to each preprocessed word because we believe that POS tags contextualise how a word is being used a sentence to differentiate different meanings [7]. This was done using *nlkt*'s POS_tag function for each token in each sentence. Each POS tag was then converted to an index. Since the same word can have different POS tags per sentence, we attached the POS tags as a one-hot vector to the word embedding index by creating a custom dataset. There are 45 different POS tags in our corpus, one way to encode this numerical data is to use one-hot encoding, the resulted encoding is not too long due to a small size of POS tag category. The one-hot POS tag encoding and other input embeddings are only concatenated when passed to the model creation to allow autograd. However, we did not yield any improvements from using this (see Evaluation Result).

2.2 Aspect 2: Semantic Textual Feature Embedding

2.2.1 Glove-twitter-25

Glove-twitter-25 is a pretrianed embedding model with embedding size of 25, which is trained on 2B tweets, 27B tokens, 1.2M vocab, and is uncased. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space [9]. We adopt glove-twitter-25 as the input embedding for our baseline model, and compare with our proposed model to evaluate on the effects on the performance by different input embeddings.

2.2.2 Word2Vec

Gensim Word2Vec models are well known for representing words as feature vectors, this allows the extraction of similarity between words, making models able to understand the words' semantic meaning better [5]. We therefore adopt Word2Vec Skip-gram model with parameters shown in table 1 to train the semantic textual feature embedding. The justification of training W2V Skip-gram model as input embedding for the in-game chat word slot filling/tagging task is as follows:

- This is a semantic embedding model with the consideration of word similarity.
- Since we train the word embedding based on training, validation and testing dataset, there will be no OOV issue for the testing purpose.

Parameters	Value
size	50 or 100
window	2
min_count	0
sg	1

Table 1: Parameters for training W2V skip-gram model

2.2.3 BERT

BERT [1] is a popular pretrained contextual word embedding, which not only contains the semantic features (e.g., word similarity) as Word2Vec, it also consider the context of a word since same word could have different meaning under different context, hence, BERT could give different word embedding for the same word based on the context of that word in the corpus.

We tried to incorporate BERT embeddings into our input embeddings but it does not fit the requirement of this in-game toxicity detection task due to the limitations below:

1. The BERT tokeniser is based on WordPiece, which is a subword-based tokenisation algorithm, spliting a word into several subwords. However, for the

NER task, the tokens are identified simply by the space, hence, additional alignment of labels and other concatenated input embeddings must be conducted as well. In addition, at training stage, the loss for the subword tokens must be considered and additional masking operation need be done when computing loss as well as at inference time.

2. Two strategies can be considered for subword tokens. We can either give the same labels to the subword tokens and compute the overall loss, or we can ignore the subword tokens and only compute and minimise the loss for the first token of a word. For the first strategy, a way to choose the predicted token at inference time need to be discussed (e.g., based on the prediction of first token or the most appeared result for first token and subword tokens). And for the latter strategy, the model could learn limited features based only on the first token.
3. The BERT tokeniser automatically split the puntuation with the letters. However, regarding the CONDA dataset, there are entities that are made up by punctuation (e.g. **::DDD**) which should be considered as one entity. But BERT will split this word and ambiguity exists when decoding BERT encoding.
4. Pretrain-then-fine-tune is a paradigm for current NLP domain. Since BERT is a pretrained model, fine-tuning on the downstream task is essential when using BERT. BERT without fine-tuning yields a moderate performance in our task since the task is conducted in a specific domain — Dota game. Since we allow the embedding matrix built from W2V to be updated during training process, it shows a significant better result than using BERT. Hence, we only adopt W2V Skip-gram embedding model as our semantic text feature embeddings.

2.3 Aspect 3: Domain Feature Embedding

Since the given train, test and validation data are based on game chats, we want to incorporate additional words meaning from glossary as the chats did not explain the meaning of the words used. This method is also used in [13], therefore we initially used the Wikipedia video games glossary. However, only 36 words out of the 11243 words in the train, test and validation data were present in this wiki glossary, hence we did not proceed with this method. As a result, we chose to use a Dota 2 game chats that is a larger dataset than the given train, test and val data. Our hypothesis is that when Word2Vec is trained on a larger dataset, higher accuracy may be achieved as it is able to have a better understand of the meaning of words in this particular context. This was done by creating Gensim Word2Vec vectors for the sentences in this dataset which allowed the concatenation with the aspect 2 vectors for the word embedding matrix. Since we believe that the given train, test and val dataset will have similar influence, we created this word embedding of size 50. If the word is not present, we concatenate with an empty vector of size 50 to make it consistent.

3 Slot Filling/Tagging Model

3.1 Stacked Seq2Seq model

Although the Transformer model is known to perform well by attending to parts of long sequences to address the memory problems in recurrent neural network models [11], it is not required for this task as the sentences in games are very short with only a few words. Therefore, we chose to use BiLSTM with CRF for the model with Dot Product Attention concatenated with a single layer of lstm_out. We tested the number of stacked BiLSTM layers in section 4.2.6. The configuration of the best model we achived on Kaggle leaderboard is specified in table 2 shown below.

Model Attribute	Parameter
Model	BiLSTM with CRF
Optimizer	SGD
Learning Rate	0.035
Dropout	0.00
Input Embedding	W2V+Domain(Dota)
Embedding size	100 (50+50)
Hidden Size	24
Batch Size	1
# Multihead	4
Attention Score	Dot Product
Attention Position	After BiLSTM + Residual
Layering	1-layer BiLSTM
Matrix	micro-F1

Table 2: Specifications of the best model (on Kaggle)

3.2 Multi-Head Self-Attention

We wrote the multi-head self-attention module used in our model. Attention outputs a sequence where each element is adjusted based on the key-query pairs from input values. The weight for adjustment is calculated by an attention score function. Self-attention is used to focus on the relation within a sequence which uses the same input as key, value and query. The formula for Attention is described below.

$$Attention(Q, K, V) = softmax(score(Q, K)) \times V \tag{1}$$

There are three attention score function considered, which are *tested* and *justified* in section 4.2.3. Three attention score functions are as followed, which are based on [10]. We used Dot Product in our best model.

$$\begin{aligned} score_1(Q, K) &= QK^T && \text{Dot Product} \\ score_2(Q, K) &= \frac{QK^T}{\sqrt{d_k}} && \text{Scaled Dot Product} \\ score_3(Q, K) &= QWK^T && \text{General-1} \end{aligned} \tag{2}$$

Multi-head attention allows attention mechanism to be run in parallel and to attend different part of the sequence from different perspectives. Different heads may learn different properties through training. Our best model used 4 heads. We also test the usage of dropout and activation functions inside our Multi-Head Self-Attention module, which turns out to be less useful, hence we abandon them. The formula for our Multi-Head module is as follows.

$$MultiHead(Q, K, V) = [head_1; \cdot; head_h] \times W^O \tag{3}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{4}$$

3.3 CRF Attachment

We test the effect of using CRF in section 4.2.7. CRF could be useful in our specific in-game chat word slot filling/tagging task since the probability of a tag following another tag is not the same and CRF layer could model this. The CRF attachment is added on the output representations after the biLSTM, Multi-Head Attention and output(hidden2tag) layer.

4 Evaluation

4.1 Evaluation Setup

We evaluated various combinations of input and parameters of our model to improve its accuracy. The accuracy is measured by using sklearn's classification report which will show the F1 score overall and for each category. The optimiser used for all testing and evaluation is *torch.optim.SGD* with learning rate of 0.03 and weight decay of 1e-4. Epoch of 2 is also adopted for all testing and evaluation. The baseline (benchmark) model is BiLSTM model with CRF and one multihead attention layer of 5 heads after one bi-directional LSTM layer. A summary of the evaluation set up for the baseline model is shown in table 3 (**N.B.**, this baseline model is used for evaluation and ablation studies, which is different from the one in section 4.2.1). In addition, except for the input embeddings evaluation, the Word2Vec word embeddings trained on training, validation and testing dataset of 100 dimension is used as emebddings for all testing and evaluation.

Our setup between different ablation studies only have one variable being changed from the proposed model to ensure that our experiments are fair.

4.2 Evaluation Result

4.2.1 Performance Comparison

We used the lab09 implementation of BiLSTM with CRF with the word embedding of Glove Twitter25 and compared the micro T-F1 score for each category with BiLSTM with CRF with the word embedding of Word2Vec Skip Gram of train, test, val data with concatenated with scaled dot product attention which

Model Attribute	Parameter
Model	BiLSTM with CRF
Optimizer	SGD
Learning Rate	0.03
Dropout	0.00
Input Embedding	W2V (Train, Test, Val)
Embedding size	100
Hidden Size	200
Batch Size	1
# Multihead	5
Attention Score	Dot Product
Attention Position	After BiLSTM + Residual
Layering	1-layer BiLSTM
Matrix	T-F1

Table 3: Evaluation setup

is our best model. The increase in validation data overall accuracy is 0.6%.
Majority of the word labels have slightly higher T-F1 scores than the baseline model, we believe that this is due to the addition of MultiHeadAttention. However, one of the reasons for using MultiHeadAttention was actually to address the problem of long sequence in the normal BiLSTM model [2], this may not be applicable to this game toxicity detection task as game chats are typically in a few words only, which may explain the small improvement in accuracy.
The increase in accuracy for the T-F1 score for T tag and D tag are much higher than other labels at (1% and 8% respectively), this is likely due to the input embedding used which was able to capture what is considered toxic in this context (for T label) and what is something used in the Dota 2 Game (for D tag). For example, there may be some game specific swear words that sound normal outside a gaming context e.g. 'killed'. Thus, it is possible that this increase in accuracy is mainly driven by the use of better input embedding.

Model	T-F1	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
Baseline	99.0826%	97.5207%	98.9553%	97.0364%	90.1333%	99.8348%	99.2467%
Proposed	99.6282%	98.3391%	99.6346%	98.8612%	98.5987%	99.9364%	99.6771%

Table 4: Performance comparison between the baseline and the proposed model

4.2.2 Ablation Study - different input embedding model

We tested the following combinations: Aspect 2 only, Aspect 1 + 2, Aspect 2 + 3, and Aspect 1 + 2 + 3. We will refer to train, test, val as TTV in the table below.

Aspect 1: POS Tagging.

This method did not yield any improvements when it was concatenated with word embedding We believe that this is due to the nature of game chats, where players then to send messages quickly, resulting in heavy abbreviations, spelling errors and lack of grammatically structure, this finding is supported by [6] as

well.
Aspect 2: Word2Vec Skip Gram of train, test and val data.
When this method was used by itself as the input embedding without other inputs in the same condition, the model achieved the highest accuracy. This suggests that the train, test and val data sufficiently captures the context of the words in the vocabulary used.
Aspect 3: Word2Vec Skip Gram of Dota game chat.
We experimented with combinations of Aspect 2 + 3 and Aspect 1 + 2 + 3, there were no improvements from only using Aspect 2 and Aspect 1 + 2 respectively. We believe that this is because the train, test and val data are already game chat data which was already of a sufficient size to understand the words' meaning. Thus it does not bring additional useful knowledge to our model despite having high coverage (10829 out of 11243 words).

Input Embedding Method	F1
Aspect 2 (W2V TTV)	99.5743%
Aspect 2 (W2V TTV) + 3 (Dota chats)	99.5833%
Aspect 1 (POS tags) + 2 (W2V TTV)	99.5833%
Aspect 1 (POS tags) + 2 (W2V TTV) + 3 (Dota chats)	99.5353%

Table 5: Ablation study for various input embedding methods

4.2.3 Ablation Study - different attention strategy: attention score

We wrote a custom *MultiHeadAttention* class based on [11] which allows us to test various attention strategies included position, attention score calculations and number of heads. BiLSTM-CRF model without attention is also tested in table 6 as a baseline. For attention score calculation testing, the forward method in *MultiHeadAttention* was modified and we tested dot product, scaled dot product and general-1 calculation, which are based on [10]. As shown in table 6, the dot product approach yields the best performance on validation set while the scaled dot product and general-1 approaches have same performance. This result is similar to the result in [10] where the authors test the effect of different attention score on Neural Question Answering Systems.

On the other hand, according to the original paper of Transformer [11], when the dimension of k d_k is large, the dot product result of Q and K will become large and its variance will be large too, which will make the gradient of the softmax function unusually small, dividing the dot product result by $\sqrt{d_k}$ will counteract this effect. However, since the hidden size used in the evaluation is only 100, with a 5 heads, the dimension of single head is only 20, which is far smaller than the dimension in [11], which means there is no need for normalising the dot product. This may be one of the resasons to justify our results.

Previous studies such as in [14] has shown the effect of initialisation in Transformer on the overall performance. Since there is an additional weight matrix used in general-1 approach, the initialisation may restrict its performance. Moreover, since only 2 epoches are allowed for training process, the weights may not be optimal. Noted that our baseline (BiLSTM-CRF without attention outperforms the other two attention score, which shows that using attention does not always work better, choosing an appropriate attention score, position and number of heads (shown in section 4.2.4 and 4.2.5) also matters.

Attention score	F1
No Attention	99.5623%
Dot Product	99.5863%
Scaled Dot Product	99.5593%
General-1	99.5593%

Table 6: Ablation study for different attention score

4.2.4 Ablation Study - different attention strategy: attention position

Three different attention positions are used for this ablation study. To test various positions, the *get_lstm_features* method for the model class is modified, we test the following positions: using attention only, attention layer before the LSTM layer and attention layer after the LSTM layer. Table 7 summarises the results of this ablation study. Using attention after the LSTM layer gives us the best result, and using attention only yields the worst result among this comparison. The justification lies on the fact that the weights in the LSTM layer is updated towards optimal during the training process, hence, the hidden output may contain more information learnt from training, resulting in a better performance than the other two positions.

Attention Position	F1
No Attention	99.5623%
Attention only	99.5323%
Attention before LSTM layer	99.5593%
Attention after LSTM layer	99.5743%

Table 7: Ablation study for different attention position

4.2.5 Ablation Study - different attention strategy: number of heads

We tried 4 different number of attention heads as shown in table 8. Using a multi-head of 5 yields the best performance. Too many or too little heads result in a moderate performance, which is consistent with the results in the original Transformer paper [11] where 8/16 heads give the best result but 1,4 and 32 heads perform worse comparing the the best number of heads. The interpretation of multihead attention is still understudied, however, there are some research trying to investigate the mechanism behind multihead attention. Authors of [12] pointed out that the most important and confident heads play consistent and often linguistically-interpretable roles. And even thought a large number of heads is used, some heads may learn the same features and only few heads which are called specialised head could learn important information. This is proved by pruning heads and they observe that specialised heads are last to be pruned. Since the dimension used in our study is small (100 dimension hidden size), applying 10 heads is too much for a small dimension, which results in a hidden size of dimension 10 for each head. Furthermore, more heads bring noise into the model, hence, using 5 heads is optimal in our model.

Number of heads	F1
No Attention	99.5623%
1 head	99.5623%
2 heads	99.5593%
5 heads	99.5893%
10 heads	99.5683%

Table 8: Ablation study for different number of heads

4.2.6 Ablation Study - different Stacked layer

Recurrent neural networks with more stacked layers are known to outperform the same model with less layers because more layers allow them to learn more complex information [4]. To test this, we can apply lstm again to the output of the previous layer of lstm including attention in the *get_lstm_features* method and compare the F1 scores for multiple categories. The results are depicted in table 9, where the trend is obvious. Using only one BiLSTM layer in our model outperforms other 2 stacked layer strategies. One hypothesis to justify this observation is overfitting. Since one BiLSTM layer could already achieve a good enough f1 score on validation dataset, stacking more layers may result in overfitting on the training dataset, which in turn decreases the f1 score on validation dataset.

Number of layers	F1
1 BiLSTM layer	99.5593%
2 BiLSTM layers	99.3014%
3 BiLSTM layers	97.8893%

Table 9: Ablation study for different stacked layers

4.2.7 Ablation Study - with/without CRF BiLSTM with CRF (conditional random field) has an extra CRF layer after the LSTM layer which is known to address labelling bias issues and eliminated unreasonable in Hidden Markov model by adopting global variance normalisation for named entity recognition tasks such as our task. Since the addition of CRF handles dependency between predicted entity names, we notice an improvement from the model without CRF. It is worth noting the limitation of CRF which is high computation for sequences of length larger than 5. Although game chats typically include less than 5 words per sentence, we still observed that BiLSTM with CRF took 10 minutes per epoch where BiLSTM without CRF took less than 2 minutes. This justification is supported by [8] for named entity recognition tasks which is a parent class of our task. As a result, the use of CRF impedes the ease of retraining when new data becomes available and hardware with higher computational power may be required when models for this is expanded to train on a larger dataset.

Model	F1
BiLSTM with CRF	99.5743%
BiLSTM without CRF	99.0706%

Table 10: Ablation study for using CRF or not

References

1. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
2. Galassi, A., Lippi, M., Torroni, P.: Attention in natural language processing. IEEE Transactions on Neural Networks and Learning Systems **32**(10), 4291–4308 (2020)
3. HaCohen-Kerner, Y., Miller, D., Yigal, Y.: The influence of preprocessing on text classification using a bag-of-words representation. PloS one **15**(5), e0232525 (2020)
4. Hermans, M., Schrauwen, B.: Training and analysing deep recurrent neural networks. Advances in neural information processing systems **26** (2013)
5. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
6. Märtens, M., Shen, S., Iosup, A., Kuipers, F.: Toxicity detection in multiplayer online games. In: 2015 International Workshop on Network and Systems Support for Games (NetGames). pp. 1–6 (2015). https://doi.org/10.1109/NetGames.2015.7382991
7. Nazeer, K.A., et al.: Part-of-speech tagging and named entity recognition using improved hidden markov model and bloom filter. In: 2018 International Conference on Computing, Power and Communication Technologies (GUCON). pp. 1072–1077. IEEE (2018)
8. Panchendrarajan, R., Amaresan, A.: Bidirectional lstm-crf for named entity recognition. In: Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation (2018)
9. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
10. Shen, Y., Lai, E.M.K., Mohaghegh, M.: Effects of similarity score functions in attention mechanisms on the performance of neural question answering systems. Neural Processing Letters pp. 1–20 (2022)
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
12. Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I.: Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. arXiv preprint arXiv:1905.09418 (2019)
13. Wang, C., Peng, X., Liu, M., Xing, Z., Bai, X., Xie, B., Wang, T.: A learning-based approach for automatic construction of domain glossary from source code and documentation. In: Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering. pp. 97–108 (2019)

14. Zhang, B., Titov, I., Sennrich, R.: Improving deep transformer with depth-scaled initialization and merged attention. arXiv preprint arXiv:1908.11365 (2019)