COMP5046 Assignment 2 Report - In-game Toxicity Detection

Group 17: Lu Liu[1] and Yunlin Peng[2]

[1] University of Sydney, NSW 2006, Australia
lliu9089@uni.sydney.edu.au
[2] University of Sydney, NSW 2006, Australia
ypen3323@uni.sydney.edu.au

1 Data Preprocessing

The Dataset is part of CONDA dataset (a CONtextual Dual-Annotated dataset for in-game toxicity understanding and detection) from the University of Sydney NLP Group. The filled slot has 7 types, which are T(Toxicity), C(Character), D(Dota Specific), S(Game slang), P(Pronoun), O(Others) and SEPA (Seperation).

Table 1. Tags in the Dataset

Tags	Training Data	Validation Data
T	4292 (4.3%)	1469 (4.4%)
C	4780 (4.8%)	1641 (4.9%)
D	1274 (4.3%)	398 (1.2%)
S	10035 (10.1%)	3322 (10.0%)
P	11997 (12.0%)	3936 (11.8%)
O	56815 (57.0%)	18985 (56.9%)
SEPA	10418 (10.5%)	3603 (10.8%)
Total	99611	33354

We transformed all datasets to lowercase and split based on space. All characters are converted to lower case for normalisation. The text is also tokenized based on space, which is to make sure that token and label is one-to-one relation.

2 Input Embedding

The input embedding from three aspects, syntactic textual, semantic textual, and domain feature embedding, are implemented, respectively, on our model. Moreover, different embedding construction methods in each aspect are implemented and compared based on the mean f1-score, and finally the best one in each aspect are selected to form different combinations All single embedding and merged embedding are compared based on the mean f1-score in order to determine the optimal input embedding for the model.

Syntactic textual embedding of each word should be constructed based on the each word's function in the sentence structure. Parts-of-Speech of each word identifies that word's function based on the way the word works in the sentence, and it defines one common type of linguistic structure, syntactic word classes (Han, 2022). Hence, the syntactic textual embedding is constructed based on the one-hot-encoding of the Parts-of-Speech of each word in the CONDA dataset. For words with the same Parts-of-Speech, they have the same one-hot-encoded embedding. Moreover, the aim of this in-game chat word slot tagging task is to identify the slot type of each word in the dota in-game chat, and words with similar slot types are highly likely to have similar POS tags, so the POS tag embedding is chosen for this aspect.

Semantic textual embedding of each word should be constructed based on the meaning and context of that word. For this aspect, we implement the methods of the pre-trained word embedding model, glove-twitter-50 and glove-twitter-100, and self-trained word embedding models, Word2Vec and FastText with CBOW and Skip-gram, and the character-level embedding to construct the word embedding of the CONDA dataset. We use glove-twitter-50 and glove-twitter-100 because as stated by Pennington, *et al.* in 2014, for different combination of embedding dimensions and corpus sizes, the glove model all has the highest semantic accuracy compared to other embedding methods. Moreover, these two embedding models are trained based on the twitter posts which are mostly informal and daily language similar as the in-game chat. Additionally, the embedding method Word2Vec with Skip-gram trains the word embedding of a word by using the word to predict its context words within a specific window size, and the embedding method Word2Vec with CBOW trains the word embedding by using its context words within the window size to predict the word itself (Han, 2022). The embedding method FastText extends the Word2Vec by considering character n-grams of each word instead of the whole word. As the result, these two embedding methods all highly consider the word's meaning, which is introduced by its context words, when constructing the word embedding, which perfectly construct semantic textual embedding. Moreover, these embedding are trained based on surrounding words, and our task of identify the slot type of each word is also highly depended on the surrounding words, so these two methods are chosen for this aspect. Furthermore, the FastText with Skip-gram is finally chosen for this aspect because it outperforms other embedding models in this aspect with the mean f1-score 0.9948 on this task.

Character-level embedding is also considered in the embedding section. Due to the abbreviations and domain-specific terms in the in-game chat dataset, most of word is unique so that out-of-vocabulary problem exists when generating embedding using word embedding. We randomly initialised embedding values for characters and update with Bi-LSTM. Model with single character embedding did not have a good performance (with mean f1-score 0.9418 on the validation set). But when combining with other embeddings, the performance boosted a lot. As the table 3 showing, when combining all types of aspect features and all types of semantic textual feature embedding (self-trained FastText word embedding

+ pretrained Glove-100 embedding + character-level embeddings), the model reached the highest mean f1 score on the validation set, which is 0.9951.

Domain embedding of each word in this task should contain the in-game chat specific information, and the embedding we used for this aspect are the external dota in-game chat and dota-specific terms. The external dota in-game chat is obtained from the in-game chat in 500000 matches in the dota 2 on the Kaggle (Anzelmo, 2019), which contains 1439488 lines of chats. After combining consecutive chats by a single user, converting to lowercase, and removing punctuation, numbers, and word length greater than 10 which are commonly useless words such as ahhhhhhhhhhhhhh, the data contains 1096996 rows. The FastText with Skip-gram, which is optimal embedding in the aspect 2, is implemented on this chat to obtain the embedding since this data is also the dota chat, and its should have similar context and structure as CONDA dataset (Weld et al., 2021), so the FastText with Skip-gram should also perform well on this external chat. This contain all in-game chat and is specific for dota, so its embedding should contain plenty of domain information. Moreover, dota-specific terminologies contains dota-specific terminologies such as gg obtained from the official dota 2 website called Dota 2 Wiki, names of materials, and names of heroes in dota games obtained from Kaggle (Anzelmo, 2019), and the one-hot encoding on these terms are performed to obtain their embedding. This is very helpful for obtaining the domain information because these terms cover most of words with the slot type of dota-specific in the dataset. Furthermore, the Dota-specific terms are chosen for this aspect because it outperforms other embedding models in the aspect with mean f1-score 0.9939 on this task.

3 Slot Filling/Tagging Model

The baseline model is Bi-LSTM CRF model, which is shown in the below figure 1.

3.1 Best Model

Our best model has several modifications based on the baseline Bi-LSTM CRF model.

Firstly, after embedding concatenation, we implemented Layer Normalization and Dropout layer with probability 0.2. Covariate shift problem, which means that gradients of weight in one layer is dependent on the output in the previous layers. Changes in output of one layer will tend to cause highly correlated changes in the inputs to the next layer. The solution is to layer is to normalize and fix the mean and variance within layers (Ba et al, 2016). Also, from the previous paper, batch normalization has a relative poor performance on RNN sequence models since it always varies in the sequence length and is sensitive to batch number. Layer normalization can solve the drawbacks of batch normalization and is effective at stabilizing the hidden state and reduce training time (Ba

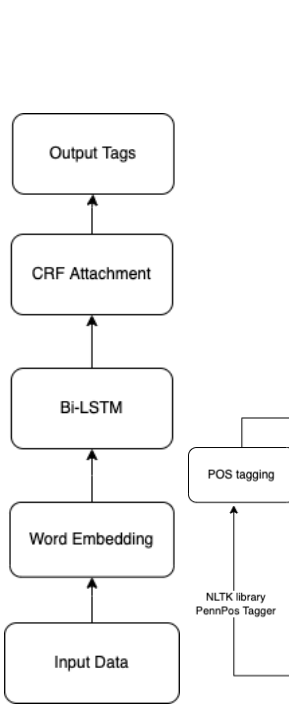

Fig. 1. Baseline

Fig. 2. Best model

et al, 2016). Dropout layer randomly de-activates neurons and helps prevent overfitting.

Then, we applied 2-layer stacked bi-lstm in our best model. Based on our assumption, more layers stacked, the model's complexity increased and thus it can better generate embeddings to represent the input data. This assumption was also supported by paper from Goldberg (2016), deep RNNs work better than shallower ones on some tasks empirically.

Moreover, self-attention layer with 8 heads was attached after stacked bi-lstm layer. Given a set of vector values and a vector query, attention is a technique to compute a weighted sum of values, dependent of the query (Han, 2022). Self-attention is to calculate the correlation between the different word in the input sequence. With splitting to several heads, the multi-head attention module performed the computation in parallel. For the attention layer position, we tried to compare attention between stacked bilstm and after stacked bilstm. From the table 5 result below, attention after all stacked bilstm has a generally better performance. Also for the attention layer score calculation, dot product, scaled dot product, cosine and general were also tested in the ablation study. Based on the final result, attention after stacked bilst with scaled-dot product score calculation has the best f1-score on the validation dataset, which also formed our best model structure.

The best model's final layer was CRF attachment. CRF layer is to handle the dependency between the predicted tags (Han, 2022). The implementation of CRF layer generate the maximum probability tags based on sentence level. As the table 7 from below ablation study, model with CRF layer had a higher F1 score than that without CRF layer.

4 Evaluation

4.1 Evaluation Setup

This experiment is conducted on the colab with GPU. The model implemented in this experiment is the Bi-LSTM with attention and CRF, and its embedding dimension of the character embedding is 256, and the word embedding dimension of Word2Vec and FastText are 100, and for the Bi-LSTM, the hidden dimension is 512, the learning rate is 0.05, the number of stacked layer is 2, and the optimizer is SGD, and for the multi-head attention, the number of heads is 8, the scoring method is scaled-dot product, and its position is after the stacked Bi-LSTM layers.

4.2 Evaluation Result

Table 2. Mean f1-scores of the best model and the baseline model. (4 d.p.)

Model	T-F1	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
Baseline	0.8859	0.5350	0.8724	0.2680	0	0.9355	0.9171
Best	0.9951	0.9804	0.9947	0.9862	0.9640	0.9994	0.9958

Performance Comparison Table 1 demonstrates that the best model proposed by us obtained the mean f1-score of 0.9951 significantly outperforms the baseline model with the mean f1-score of 0.8859. Moreover, the f1-score of each of the slot type in the best model all outperforms the one in the baseline model, especially for the dota-specific slot type and character slot type which obtain the f1-score of 0 and 0.2680 respectively in the baseline model while a f1-score of 0.9640 and 0.9862 in the best model. Thus, the best model proposed by us significantly outperforms the baseline model.

Table 3. Mean f1-scores of different input embedding models in each aspect. (4 d.p.)

Model		F1
Syntactic	POS tagging	**0.9943**
Semantic	glove-twitter-50	0.9918
	glove-twitter-100	0.9927
	Word2vec (cbow)	0.9904
	Word2vec (skip-gram)	0.9938
	FastText (cbow)	0.9920
	FastText (skip-gram)	**0.9948**
	Character embedding	0.9418
Domain	External dota chat *(1)*	0.9920
	Dota-specific terms *(2)*	**0.9939**
	(1) + (2)	0.9925

Ablation Study - different input embedding model Table 3 demonstrates that for the semantic textual embedding, the FastText with Skip-gram outperforms other embedding models in this aspect with the mean f1-score of 0.9948. This is because that FastText trains the word embedding based on the character n-grams of each word, so it can represent rare words better (Athiwaratkun et al., 2018). Moreover, the way that the Skip-gram architecture trains the word embedding via using the center word to predict its context word also enables it to represent rare words better (Mikolov et al., 2013). For in-game chat data, there are many rare words since people like to make up new words while chatting such as some variations and misspellings on words. Thus, the FastText with Skip-gram which works well for rare words has the optimal performance among all semantic textual embedding. Additionally, for the domain embedding, the embedding generated by dota-specific terms with the mean f1-score of 0.9939 outperforms the one generated by external dota in-game chat with the mean f1-score of 0.9920. Overall, the semantic textual embedding with the highest mean f1-score of 0.9948 outperforms the syntactic textual embedding with the mean f1-score of 0.9943 and the domain embedding with the highest mean f1-score of 0.9939. This might because that our task can be recognized as a kind of name entity recognition task which is proved to be enhanced by exploiting semantics (Bergamaschi1 et al., 2017).

Table 4. Mean f1-scores of different combinations of input embedding models in each aspect. (*1*, *2*, and *3* stand for aspect 1, 2, and 3, respectively, and all types in aspect 2 is the concatenation of the best model in 3 types embedding, pre-trained and self-trained word embedding, and character embedding, in aspect 2, which are glove-twitter-100, FastText with Skip-gram, and character embedding). (4 d.p.)

Combination	F1
1 + 2	0.9947
1 + 3	0.9943
2 + 3	0.9928
1 + 2 + 3	0.9950
1 + 2 (all types) + 3	**0.9951**

Besides the four combinations of the best input embedding model in three aspects, we also implement the input embedding model which concatenates the best model in all three types of embedding in the aspect 2 as the representation for the aspect 2 embedding. They are the pre-trained word embedding glove-twitter-100, self-trained FastText with Skip-gram, and the character-level embedding. Table 3 illustrates that the concatenation of all three aspects performs the best, especially the one with all types of embedding in aspect 2 since the model with three embedding concatenation has the mean f1-score of 0.9950 which outperforms all other concatenations, and the one with all types of embedding in aspect 2 even outperforms the three aspect one with the highest mean f1-score 0.9951. This is because that concatenation of different types of embedding can introduce significantly rich information into the embedding, which enhances the task performance, especially the concatenation of traditional embedding, FastText, and character-level embedding, which enhances performance on many tasks (Wang et al., 2021).

Table 5. Mean f1-scores of different attention positions with different scoring methods. (4 d.p.)

Attention positions	Scoring methods	F1
Between stacked Bi-LSTM	Scaled-Dot product	0.9936
	Dot product	0.8565
	Cosine	0.9939
	General	0.5557
After stacked Bi-LSTM	Scaled-Dot product	0.9949
	Dot product	0.9943
	Cosine	0.9947
	General	0.9937

Ablation Study - different attention strategy The model in the previous ablation study contains one Bi-LSTM block which has 2 stacked layers. In order to test different positions of the attention, we add another Bi-LSTM block with 2 staked layers on top of the current block, which results in a Bi-LSTM with 4 stacked layers in total, and the attention is tested to be placed either in between two Bi-LSTM blocks and after these two Bi-LSTM blocks. Moreover, all models in this ablation study use the optimal input embedding, the concatenation of aspect 1, 2 with concatenation of glove-100-twitter, FastText with Skip-gram, and character-level embedding, and 3. Table 5 demonstrates that the model with the attention placed after two Bi-LSTM blocks outperforms the model with the attention placed in between two Bi-LSTM blocks for all scoring methods due to the higher mean 1-scores. This might because that the purpose of the attention mechanism is to identify the important information from the hidden states of the Bi-LSTM (Han, 2022), and if place it in between two Bi-LSTM blocks, then the important information in the Bi-LSTM block on top of it cannot be identified. Additionally, for this optimal attention position, the scoring methods of scaled-dot product receives the highest mean f1-score of 0.9949. Thus, the model with attention placed after stacked Bi-LSTM wiht the scoring method of scaled-dot product outperforms all the other models in this ablation study.

Table 6. Mean f1-scores of models with different number of stacked layers. (4 d.p.)

Stacked layer numbers	F1
2	0.9951
4	0.9949
6	0.9919
8	0.9487

Ablation Study - different Stacked layer numbers Table 6 illustrates that as the number of stacked layers in the model increases from 2 to 8, the mean f1-score of the model decreases from 0.9951 to 0.9487. Thus, the model with 2 stacked layers of Bi-LSTM is the optimal one in this ablation study.

Table 7. Mean f1-scores of models with and without CRF. (4 d.p.)

Model	F1
With CRF	0.9951
Without CRF	0.9930

Ablation Study - with/without CRF Table 6 demonstrates that the model with CRF obtaining the mean f1-score of 0.9951, which outperforms the model without CRF with the mean f1-score of 0.9930. This is because that the CRF layer on top of the Bi-LSTM can handle the dependency between predicted entity names, which tests the feasible probability of the potential predicted entity sequence and selects the one with the highest probability (Han, 2022).

References

1. Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research, 57, 345–420. https://doi.org/10.1613/jair.4992
2. Ba, J. L., Kiros, J. R., Hinton, G. E. (in press). Layer Normalization. arXiv:1607.06450
3. Pennington, J., Socher, R., Manning, C. (2014). GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532–1543. https://doi.org/10.3115/v1/D14-1162
4. Han, C. (2022). *COMP5046 Natural Language Processing, lecture 6, week 6: Part of Speech Tagging* [PowerPoint slides]. Sydney School of Computer Science, University of Sydney Canvas. https://canvas.sydney.edu.au/courses/39694/pages/course-contents
5. Han, C. (2022). *COMP5046 Natural Language Processing, lecture 6, week 6: Part of Speech Tagging* [PowerPoint slides]. Sydney School of Computer Science, University of Sydney Canvas. https://canvas.sydney.edu.au/courses/39694/pages/course-contents
6. Han, C. (2022). *COMP5046 Natural Language Processing, lecture 2, week 2: Word Embeddings and Representation* [PowerPoint slides]. Sydney School of Computer Science, University of Sydney Canvas. https://canvas.sydney.edu.au/courses/39694/pages/course-contents
7. Anzelmo, D. (2019, November 14). Dota 2 Matches [Dataset]. Kaggle. https://www.kaggle.com/datasets/devinanzelmo/dota-2-matches?select=chat.csv
8. Weld, H., Huang, G., Lee, J., Zhang, T., Wang, K., Guo, X., Long, S., Poon, J., Han, S. C. (in press). CONDA: a CONtextual Dual-Annotated dataset for in-game toxicity understanding and detection. arXiv:2106.06213v2.
9. Han, C. (2022). *COMP5046 Natural Language Processing, lecture 9, week 9: Named Entity Recognition and Coreference Resolution* [PowerPoint slides]. Sydney School of Computer Science, University of Sydney Canvas. https://canvas.sydney.edu.au/courses/39694/pages/course-contents
10. Han, C. (2022). *COMP5046 Natural Language Processing, lecture 10, week 10: Attention and Question Answering (Reading Comprehension)* [PowerPoint slides]. Sydney School of Computer Science, University of Sydney Canvas. https://canvas.sydney.edu.au/courses/39694/pages/course-contents
11. Athiwaratkun, B., Wilson, A., Anandkumar, A. (2018). Probabilistic FastText for Multi-Sense Word Embeddings. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1–11. https://doi.org/10.18653/v1/P18-1001

12. Mikolov, T., Chen, K., Corrado, G., Dean, J. Efficient Estimation of Word Representations in Vector Space. In 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, 2013.
13. Bergamaschi, S., Cappelli, A., Circiello, A., Varone, M. (2017, June). Conditional random fields with semantic enhancement for named-entity recognition. Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics. https://doi.org/10.1145/3102254.3102286
14. Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., Tu, K. (in press). Automated Concatenation of Embeddings for Structured Prediction. arXiv:2010.05006.