Assignment 2

Zedong Wen and Jielin Feng

University of Sydney, NSW 2006, Australia
zwen8150@uni.sydney.edu.au
jfen8896@uni.sydney.edu.au

1 Data preprocessing

'No Content'

2 Input Embedding

Through the input embedding section, we used synthetic texture embedding, semantic feature embedding, and domain feature embedding. Among them, PoS tagging and dependency path are used for syntactic texture embedding, word2vec's skipgram is used for semantic feature embedding, and 3-letter-suffix is used for domain feature embedding. First of all, PoS tagging, dependency path, word2vec's skip-gram, and 3-letter-suffix are designed to experiment with each separately.

For PoS tagging, we first give the part of speech corresponding to each word in its corresponding sentence, that is, give its corresponding grammatical attributes according to the context. Grammatical attributes are the grammatical tag of a word, which are categorical variables. To transform the word-based category variable into the vector and facilitate the subsequent embedding method to do further concatenation, we need to increase the dimension of the category variable. We decided to use word2vec to further increase the dimension. Both PoS and dependency have dependent relationships, so Word2Vec can be used to find similar relationships on the premise of dimension upgrading, and the vector feature is extracted from the tagging of PoS.

For dependency pathing, we first give the dependency relationship of each word in each sentence in the sentence, that is, give its corresponding dependency grammar tag according to the context. The dependency grammar tag is also a single word attribute, which is a categorical variable. Like POS tagging, we also put the dependency grammar tag obtained after using dependency parsing into word2vec. The reason for this is to convert the category variable into a high dimensional vector and extract the features of the tag of the dependency grammar.

Both PoS tagging and dependency pathing belong to syntactic embedding. The reason why we use syntactic textual embedding in this task is that our task is to label each word in the corresponding sequence. Therefore, it is very important to understand the relationship between words. PoS and dependency can tag part of speech according to the context in the sentence. Putting the part of speech tag and dependency grammar tag into Word2Vec can further find the similarity relationship between words with different parts of speech according to the context. Therefore, we can better extract the features of the relationship between words. At the same time, syntactic embedding can solve the ambiguity problem that semantic textual embedding cannot solve to some extent. When semantic embedding cannot distinguish the same word with different semantics, syntactic embedding can add a new dimension feature, that is, grammar attribute, to better distinguish them.

For the Word2Vec (Skipgram), its meaning is to find the semantics corresponding to each word in each sentence and represent it with a vector of 100 dimensions. Skip-gram predicts the surrounding word based on the center word. Since the sentences in this data are all dialogue information records in the game, the lengths are relatively short, so we set the window size to a small value of 3. 3 specifies that the number of surrounding words predicted by the center is 3. A small window size results in a similarity score that indicates that the words are interchangeable. Word2Vec belongs to the method of Semantic feature embedding. The reason we use it is that the central idea of Word2Vec is that words with the same neighbors contain similar semantics, and in skip-gram, for different input words, if they have the same output, then It means that the phrases used as input have high similarity. Therefore, using Word2Vec skip-gram can well extract the vector features expressing the similarity of different words.

For domain feature embedding, we use the 3-letter-suffix method. 3-letter-suffix is constructed by keeping only the last three letters of each word. If the word length is less than 3, use "-" to indicate. Then, just like the syntactic and semantic feature embedding mentioned above, the word-based features that are generated by 3-letter-suffix are put into word2Vec and converted into vector features. The reason we use the 3-letter-suffix method is that the suffix of a word usually contains the most information, especially for some place names, slang words, etc., their suffixes may show a kind of trend. Therefore, we hope that the embedding method that retains the last three letters can further represent different words.

All of the above embedding methods ultimately require high-dimensional vector feature conversion through word2Vec. We uniformly set the dimension size to 100 and the window size to 3. The reason for setting 3 is mentioned above. The reason why all the dimension sizes are set to 100 is that we want to extract the feature information of different words in more detail, so a dimension size of 100 can unify different embedding methods when the dimension is high. Also, set all dimension sizes to 100 to keep dimension consistency for all embedding methods.

Each of the feature embeddings mentioned above has certain disadvantages if used alone. For example, semantic feature embedding cannot distinguish the same word with different semantics. In this specific dataset, since the training data are all chat content in the game, many phrases do not have dependencies or correct grammatical relationships, so syntactic textual embedding may not produce very good results. It may be difficult for 3-letter-suffix to extract suitable features through the suffix information due to the particularity of game chat phrases. In order to solve the possible drawbacks of different single feature embeddings, we have created different combinations of feature embeddings, hoping to expand the vector feature of each word by combining different embedding methods, so that it has more accurate features to achieve a better f1 score.

Combination 1 we created is to combine PoS tagging and dependency path. We horizontally combine the extracted PoS tagging vector feature and dependency path vector feature, that is, convert the original 100-dimensional features into 200-dimensional features. The reason we do this is that PoS tagging and dependency path belong to syntactic feature embedding. We hope that combining the two methods can better extract part-of-speech features according to the context, resulting in better performance than a single method.

The combination we created 2 is to combine Word2Vec and 3-letter-suffix. Its construction principle is also to perform horizontal combination and expand the feature dimension to 200. The reason for this combination has been mentioned above. We think that Syntactic textual embedding does not necessarily produce good results in our dataset, because in the chat sentences, the grammaticality and dependencies between words are slightly weak. Therefore, we expected that not adding syntactic textual embedding may have a good impact on the final performance.

Combination 3 we created is Word2Vec plus 3-letter-suffix plus PoS tagging. The construction method is still a horizontal combination, forming the feature of dimension 300. The reason for this is that we believe that in each sentence on this dataset, word-to-word dependencies may be weaker than common grammatical properties. Therefore, it is expected that the final performance may be improved by excluding dependencies.

The combination 4 we created is to combine all the embedding methods, that is, the vector features produced by all the methods of horizontal combination finally give each word a 400-dimensional vector feature. The reason for this is that, as mentioned above, each single embedding method may have certain drawbacks. So we want to solve problems where individual methods have limitations by combining all the methods. And 400 is the highest feature dimension of all combinations. This represents a very detailed feature description for each word, and we hope that this way of combining all the embeddings can improve the final performance to a certain extent.

3 Slot Filling/Tagging model

3.1 Model Design

Based on the Baseline Model, that is, the Bi-LSTM CRF model, and referring to the existing literature [1][2][3], we have come up with the design scheme of the Model. Our model consists of the following parts: input layer, embedding layer, n x Bi-LSTM layers, attention layer, and finally CRF output layer. In the forward propagation process, the output of each layer is used as the input of the next layer. where n in n x Bi-LSTM layer represents n stacked layers. The attention layer adds an attention mechanism. The CRF output layer changes the original softmax output of Bi-LSTM and adds the maximum possible output obtained by combining the label context after CRF. For the specific design principles and construction process, please refer to the following content in section 3

3.2 Stacked Seq2Seq model

The design of Stacked Bi-LSTM is that the Stacked Bi-LSTM model includes: input layer, embedding layer, n x Bi-LSTM layers and output layer. The principle of Stacked LSTM is to construct multiple hidden LSTM layers, and each LSTM layer contains multiple memory cells. Stack LSTM can make the network deeper, enabling predictions for more complex problems. The study [4] points out that deepening the network by stacking LSTM layers will achieve better performance than simply adding memory cells. The article [3] also confirms that by constructing multiple hidden LSTM layers, Stacked LSTM has better performance than ordinary LSTM. Therefore, we added the Stack parameter to the original BiLSTM architecture to control the number of hidden LSTM layers. The settings of Stack are 0, 1, and 3. 0 means no hidden LSTM layers, 1 means 1 hidden LSTM layer, and 3 means 3 hidden LSTM layers. It is expected that a larger stack value can increase the network depth, increase the complexity of the model, and thus bring better performance in the final performance.

3.3 Attention

The article [5] proposes the idea of attention-LSTM. The most important information may exist anywhere in the sequence, but it often cannot be accurately extracted. This problem is solved by Attention-based Bi-LSTM. It can extract the important information in the sentence. The Bi-LSTM after adding the attention mechanism often behaves better compared to the ordinary Bi-LSTM model. This is because Attention calculates the attention probability distribution value assigned to each word in the sequence, which can effectively extract the important keywords in the sequence to a certain extent. Extracting important keywords can play an important role in the semantic representation of the text, the enhancement of semantic representation is very likely to get better performance in a tagging task. Thus we expect that adding an attention mechanism will lead to better performance of our model. We have designed three Attention Methods, which are dot products, Scale dot products, and Content-based with the formula $score(s_t, h_i) = {s_t}^T h_i$, $score(s_t, h_i) = \frac{{s_t}^T h_i}{\sqrt{n}}$ and $score(s_t, h_i) = cosine[s_t, h_i]$, respectively. Based on the existing research [6], we learned that adding an attention layer between the Bi-LSTM layer and CRF layer is widely recognized. Therefore, we set the attention position between the Bi-LSTM layer and the CRF output layer. As shown in Figure 1, the attention layer is positioned in the “d” part.

3.4 CRF Attachment

When using Bi-LSTM alone, because it only focuses on the relationship of text context and lacks the conditional constraints on label transfer, the label sequence of the model may be wrong. The CRF can effectively avoid this problem. When combining Bi-LSTM and CRF, the softmax output layer of Bi-LSTM is independent of each other, it only outputs the maximum possible result information learned from the context, and adding a CRF with a transfer feature function behind it can be used to learn the label context at the same time. Jurafsky [7] mentions that CRF is a log-linear model that given all sequence input, the output is the assigned label probability. It allows the traditional Bi-LSTM to learn the order between labels while considering the text context. After adding CRF, it no longer simply outputs the maximum probability label through the softmax layer of Bi-LSTM but considers the relationship of the label chain, so that the final tagging has the best score under the premise of satisfying the label rules. In this way, the phrase labels in the predicted sequence can be limited to a certain order, and many wrong sequence labels can be avoided.

We realize the combination of Bi-LSTM and CRF by constructing the decoding algorithm of HMM, that is, the Viterbi algorithm. The Viterbi algorithm is a dynamic programming algorithm whose goal is to find the optimal path. The Viterbi algorithm can get the optimal probability output of the sequence. After the output of the original LSTM, use the Viterbi algorithm to calculate the probability score of each combination, and find the maximum score, that is, the label output with the maximum probability. Since CRF further judges the rationality of the label order based on the traditional LSTM only using the softmax layer to obtain the maximum probability label and eliminates the label combination that does not conform to the rules. We hope that the model with the added CRF can achieve better performance.

4 Evaluation

4.1 Evaluation setup

Data

Datasets are from part of the CONDA Dataset. This dataset is from USYD's NLP Group and focuses on tagging from in-game chat. It includes a training set, validation set, and testing set. Datasets contain 7 types of labels, T (Toxicity), C (Character), D (Dotaspecific), S (game Slang), P (Pronoun), O (Other), and SEPA (SEPA). The training set contains 26078 data samples. It has sents and labels. Sents contain different in-game chat sentence groups, and labels tag words in the corresponding sequence. The Validation set contains 8705 data examples, with both sentences and labels. The testing sets contain 500 testing samples, which only contain sents and do not contain labels.

Baselines

The baseline model used in this study is the Bi-LSTM CRF model. It uses an SGD optimizer, learning rate equals 0.01, weight decay equals 1e-4, and epoch is set to 2. Hidden dimension equals 50.

Implementation Details

For all other Bi-LSTM models used in this study. All of them use the Adam optimizer, learning rate equals 0.001. epochs are all set to 2. hidden dimension equals 50. All experiments were run on Colab. The operating environment is GPU T4, and the RAM is 12.68 GB.

Evaluation Metrics

For the baseline Model and all other Bi-LSTM experimental models, the metrics used for evaluation are T-F1 (Exclude O), T-F1 (T), T-F1 (S), T-F1 (C), T-F1(D), T-F1(P) and T-F1(O). In this experiment, the reason why T-F1 does not include O is that the occurrence frequency of O is too high, which will affect the results of other types of overall F1. Which is referenced from [8].

4.2 Evaluation result

Performance Comparison

Model	T-F1	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
Baseline Model	0.9599	0.9176	0.9551	0.9122	0.6784	0.9914	0.9893
Final Model	0.9848	0.9558	0.9825	0.9689	0.9246	0.9962	0.9845

Table 1: slot labeling performance on CONDA for the baseline model and the best final model. Best Final Model is the Bi-LSTM with Embedding (PoS+Dep+3Letter+W2V) and Attention (Content-based)

Compared with the Baseline Model, which is a BiLSTM +CRF model, our best model, Bi-LSTM With Embedding and Attention has a significant improvement, which increases the f1 score from 0.9599 to 0.9848. The Best Final Model has more improvement than the Baseline Model in all categories except for O. Among them, the improvement of the D category is the most obvious, and the Best Final Model directly improves the f1(D) from 0.6784 to 0.9246. Adding Embedding, Attention, and removing CRF significantly improves the performance of the model. The specific analysis will be carried out below.

Ablation Study- different input embedding model

Combinations	T-F1	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
P	0.9810	0.9510	0.9819	0.9573	0.8643	0.9954	0.9999
D	0.9790	0.9530	0.9801	0.9598	0.7990	0.9947	0.9999
W	0.9820	0.9558	0.9798	0.9604	0.8920	0.9952	1.0000
L	0.9825	0.9537	0.9807	0.9610	0.9045	0.9957	0.9997
P+D	0.9816	0.9496	0.9822	0.9567	0.8894	0.9957	0.9999
W+L	0.9808	0.9517	0.9816	0.9586	0.8518	0.9957	0.9999
W+L+P	0.9816	0.9530	0.9801	0.9610	0.8794	0.9957	0.9999
W+L+P+D	0.9829	0.9544	0.9825	0.9603	0.8995	0.9962	1.0000

Table 2: Ablation studies comparing different input embedding In the proposed model. we introduced four aspect embeddings, including PoS tagging(P), Dependency path (D), Word2Vec Skipgram (W), and 3-letter-suffix (L). The ablation testing is conducted in a different combination of aspect embedding from each higher-level of aspect groups. The highest performance is highlighted in green, the lowest is marked in red.

As can be shown in Table2, The Combination of W+L+P+D produced the best overall performance with an F1 score of 0.9829. It is worth noting that the embedding method of Single 3-Letter-Suffix achieved the best overall performance in a single embedding The total F1 score is second only to The Combination of W+L+P+D. And 3-Letter-suffix has the highest F1(D) score of 0.9045, which is the only embedding method with D's f1 above 0.9. Using Dependency alone as the embedding obtained the worst performance, and its f1 was only 0.979. The performance of PoS tagging alone was not very satisfactory, and the f1 score was only 0.9810. However, the combination of P+D obtained better results than any single embedding method. with an f1 score of 0.9816. The result of Word2Vec alone is rather mediocre, and the f1 score is 0.9820. The combination of W+L and W+L+P did not achieve the expected results, with an f1 score of 0.9808 and 0.9816, respectively.

The reason why using Dep and PoS alone is not ideal is related to the specific datasets. The data are relatively short in-game terms, and many sequences are only composed of different short words, game terms, or abbreviations of game terms, and there is no strong logical relationship, so it is difficult to find the part of speech corresponding to the word according to the context, that is, the grammar label and the dependency grammar tag. So using Dep and PoS alone is not ideal. However, it can be seen that after the combined use of Dep and PoS, f1 is significantly improved, and the overall improvement of f1 is mainly due to the more accurate identification of the D category. This may be because Dep and PoS are not effective individually, but after combined use, the D-category words that are difficult to identify are given richer and more complete part-of-speech features, thus leading to the improvement of the uncommon category f1, realizing the overall f1 improvement.

Using the 3-letter-suffix embedding method alone has the best D-category f1 score and the total f1 is second only to the full combination. This may be because the D category represents Dota-specific words, which may be abbreviations or the names of game items, actions, and characters. The suffixes of these words are likely to have a certain trend or some similar characteristics. Therefore, adding the suffix feature will make it achieve more accurate identification.

The Combination of W+L+P+D has the best results, compared to other combinations, The Combination of W+L+P+D has a slight improvement in the F1 score of each category and the total f1 has been promoted. This confirms my conjecture in the section2 embedding input. The final feature matrix with 400 vector features produced by combining all embedding methods solves the limitation of a single embedding method and gives words with more accurate and rich feature information. Hence produce the best performance.

Ablation Study- different Stacked Layer

Num of Stacked layer	T-F1	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
0	0.9829	0.9551	0.9822	0.9610	0.8920	0.9964	0.9995
1	0.9748	0.9408	0.9765	0.9512	0.7538	0.9952	0.9954
3	0.9266	0.8400	0.9383	0.9037	0.0000	0.9850	0.9955

Table 3: Ablation studies comparing different Stacked Layers In the proposed model. The highest performance is highlighted in green, the lowest is marked in red.

As can be shown from Table 3, the increase of Stack layers did not lead to better results. Bi-LSTM without stacked-layer has the best performance with an f1 score of 0.9829. When the number of stacked layers is equal to 1, there is a slight drop in the total f1 score. When the number of stacked layers is equal to 3, the total f1 score has a very significant drop, and the f1 score of the D category drops directly to 0. This shows that in our study, increasing the network depth to improve the model complexity will reduce the model performance. This can be caused by neither network degradation nor over-fitting. The article [9] shows that when the network is too deep, as the network increases, the accuracy of the training set will be too saturated, and then it will drop rapidly. But this drop is not caused by overfitting. The over-fitting phenomenon is because the network complexity is too high, but the number and complexity of our actual data samples cannot match the model complexity, so it will lead to over-fitting. Which performs in the validation set is much worse than that of the training set. In our situation, when the stacked layer is equal to 3 when running the first epoch, the train f1 is 0.8491, and the validation f1 is 0.8486. When running the second epoch, the train f1 is 0.9531, and the validation f1 is 0.9468. It can be seen that the performance of the training set is not better than that of the validation set, so the possibility of overfitting is ruled out, and the reason for the poor performance of the stacked layer is the network degradation phenomenon. Therefore, based on this dataset, when the stacked layer is equal to 0, that is, when no stacked layer is added, the model performs best.

Ablation Study- different attention strategy

Attention method	T-F1	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
Scale dot product	0.9839	0.9564	0.9816	0.9610	0.9246	0.9969	0.9908
Dot product	0.9847	0.9564	0.9825	0.9677	0.9246	0.9962	0.9889
Content based	0.9848	0.9558	0.9825	0.9689	0.9246	0.9962	0.9845

Table 4: Ablation studies comparing different attention strategies In the proposed model. The highest performance is highlighted in green, the lowest is marked in red.

As can be shown in Table 4, Compared with the previous Bi-LSTM models without attention, Bi-LSTM with attention has significantly better performance. All three Attention methods improve the overall f1 score. Among them, Content-based has the best performance with a total f1 of 0.9848. the dot product method has a similar performance to the content-based method with an f1 score of 0.9847. The f1 score of the scale dot product method is slightly lower which is 0.9839. This confirms that the attention mechanism mentioned in the article [5] can find important information in the sequence, thereby improving the performance of the model. It is worth mentioning that after adding the attention mechanism, the three attention methods have improved the performance of F1 (D) significantly. This may be because the semantic information of the text is improved by the finding of the important keywords in the sequence. For the D category that contains Dota game terminology, more accurate semantic information increases the accuracy of the judgment of this category, and thus improves the overall performance.

Ablation Study- with/without CRF

CRF	T-F1	T-F1(T)	T-F1(S)	T-F1(C)	T-F1(D)	T-F1(P)	T-F1(O)
No	0.9848	0.9558	0.9825	0.9689	0.9246	0.9962	0.9845
Yes	0.9845	0.9558	0.9825	0.9665	0.9246	0.9962	0.9899

Table 5: Ablation studies comparing with or without CRF In the proposed model. The highest performance is highlighted in green, the lowest is marked in red.

As can be shown in Table 5, adding CRF did not improve the performance of the model. In general, with or without CRF does not affect the performance of the model. The article [7][10] mentioned that CRF will learn the context of the label based on Bi-LSTM's learning of the context of words. And a certain order is limited to the phrase labels in the detected sequence. In our experimental results, adding CRF did not achieve good results. This may be because the data in our dataset are all game phrase sequences, the combination order of the labels is relatively random, and a very reasonable label context relationship is not obtained. in which is in this study, the output label corresponding to the sequence input has no specific order rules. Even the label chain calculated by CRF may not match the real label order in the test set. Therefore, after adding CRF, the final model effect cannot be improved and even slightly decreased.

References

1. Luo, L., Yang, Z., Yang, P., Zhang, Y., Wang, L., Lin, H., & Wang, J. (2018). An attention-based BiLSTM-CRF approach to document-level chemical named entity recognition. *Bioinformatics*, *34*(8), 1381-1388.
2. Chen, T., Xu, R., He, Y., & Wang, X. (2017). Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. *Expert Systems with Applications*, *72*, 221-230.
3. Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.
4. Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645-6649). Ieee.
5. Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., & Xu, B. (2016, August). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 207-212).
6. Wang, Z., & Yang, B. (2020, August). Attention-based Bidirectional Long Short-Term Memory Networks for Relation Classification Using Knowledge Distillation from BERT. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)* (pp. 562-568). IEEE.
7. Jurafsky, D., & Martin, J. H. (2020). Sequence labeling for parts of speech and named entities. *Speech and Language Processing*.
8. Weld, H., Huang, G., Lee, J., Zhang, T., Wang, K., Guo, X., ... & Han, S. C. (2021). CONDA: a CONtextual Dual-Annotated dataset for in-game toxicity understanding and detection. *arXiv preprint arXiv:2106.06213*.
9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
10. Chen, T., Xu, R., He, Y., & Wang, X. (2017). Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. *Expert Systems with Applications*, *72*, 221-230.

Appendix

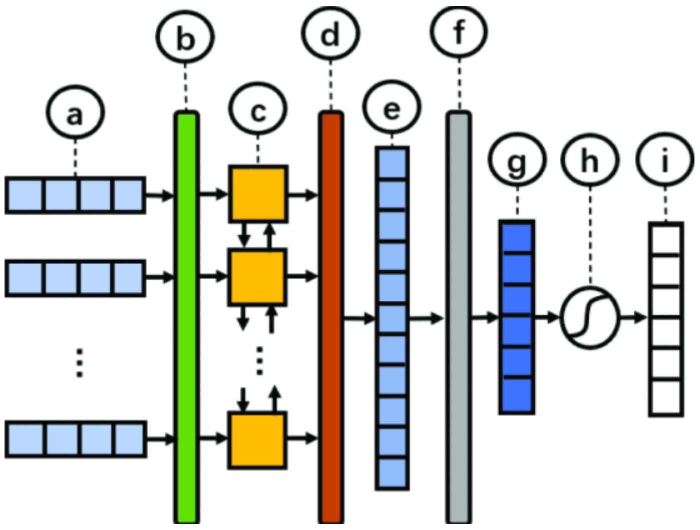

Fig.1 Attention – BiLSTM model structure citing from [6]