ANLP Assignment 2 (2021)

Student 1: s2153223 Student 2: s2168463

1. Introduction

In this assignment, we explore some mistakes the contextualized and static embedding taggers make, and how different combination functions affect the tagging result. Furthermore, we try to improve the performance of tagger by combining two embedding method.
Through the analysis on the wrong tagging, static embedding tagger is incapable of handling words having more than one tag because each word only corresponds to one embedding vector, leading to the same prediction from classifier. As for contextualized embedding tagger, it provides more accurate result (0.958 f1-score using reduced mean combination function). It summarizes the information of context to represent the words, but also is disturbed by context sometimes. Besides, combination function which drops some information, may also lead to wrong prediction.
At last, by merging static and contextualized embedding together, the performance of tagger improves slight with f1-score reaching 0.963.

2. Methodology

The tagger applied in this assignment can be divided into two parts: word embedding and classification. Word embedding is the first task of the whole model. Two different word embedding methods are discussed in this experiment:

- Static word embedding:
  The static word embedding is gained from GloVe. It contains the representation of a large number of words, obtained by applying an unsupervised learning algorithm training on global world-world co-occurrence counts [2]. Each time a word is extracted from the sentence, the GloVe always produce the same word vector to represent those words, namely static embedding.
- Contextualized word embedding:
  The contextualized word embedding applied in this experiment is based on BERT, a pre-trained transformer model consisting multiple encoders [1]. To get the embedding, firstly the sentences are tokenised basing on WordPiece embedding, and converted into a index list. And we choose the output of hidden layer as contextualized word embedding of tokens.The representation is dynamically varying as the change of context.

The next step is classification. The resulted embedding vectors are fed into a multinomial logistic regression classifier. It is trained to map the representation of one word into a tag.

3. Experiment and result

3.1 Effect of contextualized and static word embedding on results

We test Static embedding and contextualized (combination function uses *REDUCE_MEAN*) embedding taggers on test set and the results have been given in Table 2 and Table 3. From the result, contextualized embedded tagger works better, but both types of taggers make mistakes. Table 4 lists some prediction mistakes that one tagger makes and the other does not.

- Static embedding tagger:
  By analyzing the error it makes, the static embedding tagger is incapable of tagging the words that have same spelling but different POS tags. The reason is that the static embedding creates a fixed vector representation for each word, and does not contain context related information. The classifier always receives the same vector, therefore, it always produces the same prediction. Because of that, its performance is supposed to be similar to the baseline classifier which always

chooses the most frequency tags, as shown in Table 2 and Table 1.

- Contextualized embedding tagger:
  As shown in Table 3, this tagger has much better performance compared to static one. From the result, contextualized word representation may lead to wrong prediction on words which shows in similar place but with different POS tags in the sentence. As the embedding method represent the words based on the its context, similar contexts lead to similar word embedding and result similar predictions from classifier. For example, the 'ing' form of a verb that follows a auxiliary or is used to describe one component in sentence appears in similar place as adjective. The classifier may assign wrong tag to the verb. Table 4 shows some example of wrong tagging.

Generally speaking, static embedding tagger cannot tag the words with more than one tags correctly, but the training speed is fast. Contextualized embedded tagger has a good performance and solves the problem of static embedding, but sometimes it will be disturbed by the context. Besides, it needs to calculate word embedding dynamically which is time-consuming compared to static method.

3.2 Combining static and contextualized word embedding

From previous analysis, static word embedding does not take context into consideration, while the contextualized word embedding only relies on context, even being disturbed by it. As a result, we try to merge those two methods together to utilize their advantage and compensate each others' drawback. In experiment, we generate a new representation of words by stacking static embedding directly after contextualized word embedding (using REDUCE_MEAN). Then train our classifier basing on the new embedding. From result, the new representation improves the performance of tagging (0.963 accuracy and weighed average f1-score), shown in Table 5. The extended features contain both word itself and context information, contributing to the classification. However, this new features need both static and contextualized word embedding, which is more time-consuming. And the longer representation leads to more parameters in classifier, increasing the computational cost and further extending training time.

3.3 Effect of combination functions on results

We generate the embedding of the word by combining the vectors of its sub-words through different combination functions and test it on test set. The result show that the performance of five different functions are very close in terms of precision, recall, f1-score and accuracy and REDUCE_MEAN function works best, as shown in Table 6. By observing tagging result, those functions leads to some different errors. Table 7 lists some ambiguous tokens and their tags.
The FIRST and LAST TOKEN function directly choose the vector of one token as the representation of the whole word, which may lead to bias on meaning. For example, in sentence (1), the word 'beheading' is tokenized into 'be' and 'heading'. The first token function choose 'be' as the representation of the word, and it may distinguish with the word 'be', leading to a wrong prediction of 'AUX'. Similarly, as for word 'playin' and 'symptomatic', representing the whole word by one of the token could lead to error.
REDUCE_SUM and REDUCE_MAX function have an advantage in predicting words consisting of multiple tokens. As for sentence (5), 'Bhatia' is a person's name, which is tokenized into 'B', 'hat' and 'ia'. In this case, the word is divided into too many parts, a single token can not represent the whole word. On the contrary, the correct prediction is obtained after considering all vectors.

Overall, generally, all five combination function performance present good result on test set. FIRST and LAST TOKEN function discards part of information in sub-words, which leads to a little bias on the representation. REDUCE_MEAN and REDUCE_MAX summarizes characteristic from all sub-words, while may weaken the influence of the key part.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

Appendix: Tables and Figures

	precision	recall	f1-score	support
NOUN	0.674	0.929	0.781	4136
PRON	0.968	0.933	0.950	2158
PUNCT	0.994	0.985	0.990	3098
VERB	0.884	0.809	0.845	2640
accuracy	/	/	0.858	25098
macro avg	0.841	0.780	0.792	25098
weighted avg	0.872	0.858	0.855	25098

Table 1: Classification report for top tag classes of baseline classifier on test set (Result of Preliminary Task)

	precision	recall	f1-score	support
NOUN	0.825	0.869	0.846	4136
PRON	0.965	0.926	0.945	2158
PUNCT	0.994	0.983	0.988	3098
VERB	0.833	0.821	0.827	2640
accuracy	/	/	0.867	25098
macro avg	0.816	0.795	0.797	25098
weighted avg	0.868	0.867	0.866	25098

Table 2: Classification report for top tag classes of static embedding

	precision	recall	f1-score	support
NOUN	0.934	0.944	0.939	4135
PRON	0.994	0.993	0.993	2156
PUNCT	0.994	0.991	0.992	3096
VERB	0.966	0.966	0.966	2638
accuracy	/	/	0.958	25082
macro avg	0.933	0.937	0.935	25082
weighted avg	0.958	0.958	0.958	25082

Table 3: Classification report for top tag classes of contextual embedding(reduce mean)

Token	Tag(static)	Tag(contextualized)	Tag
(1)via Microsoft **Watch** from Mary	VERB	PROPN	PROPN
(2)wise enough to **watch** his step	VERB	VERB	VERB
(3)a **little** nostalgic	ADJ	NOUN	ADJ
(4)entering a **floating** world of dreams	VERB	ADJ	ADJ
(5)keeping **required** levels available	VERB	ADJ	VERB

Table 4: Different results of static and contextualized embedding.(An underlined tag indicates a prediction error.)(Contextualized embedding used REDUCE_MEAN function.)

	precision	recall	f1-score	support
NOUN	0.942	0.947	0.945	4135
PRON	0.997	0.995	0.996	2156
PUNCT	0.997	0.992	0.995	3096
VERB	0.968	0.969	0.968	2638
accuracy	/	/	0.963	25082
macro avg	0.938	0.944	0.941	25082
weighted avg	0.963	0.963	0.963	25082

Table 5: Classification report for top tag classes of classifier basing on merged word embedding

	first token	last token	reduce max	reduce mean	reduce sum
accuracy	0.954	0.957	0.957	0.958	0.957
precision(weighted)	0.953	0.957	0.957	0.958	0.957
recall(weighted)	0.954	0.957	0.957	0.958	0.957
f1-score(weighted)	0.953	0.957	0.957	0.958	0.957

Table 6: Classification report for combination functions.

Token	(a)	(b)	(c)	(d)	(e)	Tag	Tokenization
(1)two **beheading** videos	AUX	AUX	NOUN	NOUN	NOUN	NOUN	'be','heading'
(2)It's **symptomatic**.	NOUN	ADJ	NOUN	ADJ	NOUN	ADJ	'sympt','omatic'
(3)thanks 4 **playin**	VERB	ADP	NOUN	VERB	NOUN	VERB	'play','in'
(4)a **fledged** system	VERB	VERB	ADJ	VERB	ADJ	ADJ	'fled','ged'
(5) **Bhatia**(in a name)	PROPN	X	PROPN	X	PROPN	PROPN	'B','hat','ia'

Table 7: Taggers of combination functions differ in predicting tag for the same word. (a):first token,(b):last token,(c):reduce max,(d):reduce mean,(e):reduce sum.(The token to be predicted is bold. An underlined tag indicates a prediction error.)