ANLP Assignment 2 2023

Marked anonymously: do not add your name(s) or ID numbers.

1 Logistic regression tagging with word embeddings

To compare the performance of the most common tag (MCT) and logistic regression models, we examine their categorisation reports of the validation set. The logistic regression model achieves 17–27% higher micro, macro, and weighted average scores of precision, recall, and F1 than the most frequent tag model. To find out the underlying reason for such improvements, we analyse the samples containing the 'B-kids' BIO tag due to their major increase of 62%, 83% and 71% in precision, recall and F1 scores, respectively.

The words in the validation set where the actual BIO annotation label is 'B-kids' are '1', '8', 'one', 'two' and 'three'. The full report reveals that the most frequent tag model never predicts the 'B-kids' tag, which can be attributed to the following factors. The words' one', 'two' and '1' occur more frequently in the training set with other BIO tags. Hence, the probability of the B-kids tag is lower than that of other tags. Additionally, the word '8' never appears in the training set with the 'B-kids' tag. Therefore, the probability that the word has the respective tag is zero. Lastly, even though for the word 'three' the greatest probabilities are both 'B-kids' and 'B-date_period', 'B-kids' is not selected since the first tag in the probability array is 'B-date_period' due to tags being stored in alphabetical order.

By examining the full report for the logistic regression model, we observe that the model correctly predicted the tag five times out of six times when the true BIO tag annotation label was 'B-kids', outperforming the most frequent tag model. This can be attributed to the logistic regression model being trained on token embeddings, which are computed using the word2vec algorithm. Tokens that have similar tokens in their context windows have similar encodings. Specifically, the tag was accurately assigned for words that had terms like 'child', 'children', 'kids' or 'sons' in a two-token window. Their embeddings closely resembled those of words associated with the 'B-kids' tag in the training set, which featured the same terms in their context windows. Furthermore, the logistic regression model can correctly assign tags to unseen words ('8') since such words will have similar embeddings to words from the training set due to the similar contexts.

Figure 1 illustrates a conceptual simplification of the decision-making process of the logistic regression model. One can observe that the embeddings of token 'three' with tags 'B-kids' and 'B-date_period' from the training are distinctly separated due to dissimilar context windows. This allows the decision boundary to clearly separate the training data points and correctly classify an unseen validation set embedding.

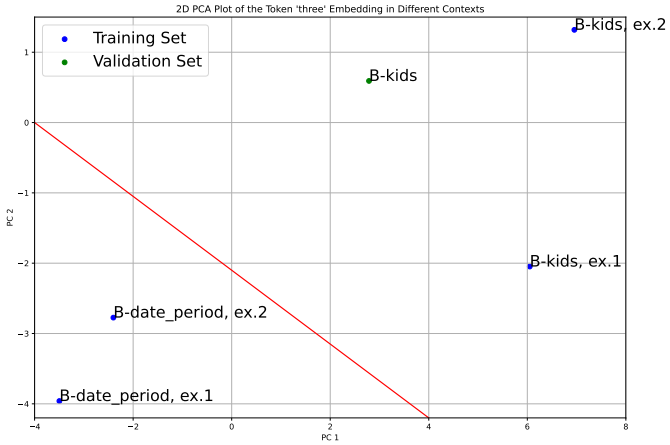

Figure 1: Visualisation of the word 'three' embedding when labelled with 'B-kids' and 'B-date_period' tags, obtained using PCA. The red line illustrates a potential decision boundary.

2 BIO tagging for slot labeling

The provided algorithm that enforces the constraint of an 'I' tag being followed by a 'B' or an 'I' tag of the same type first generates a list of plausible tags for a token produced by the model. Since the labels are arranged from most probable to least probable, it assigns the token's label to be the first on the list. However, in the case when the most likely label starts with an 'I' and is not of the same type as the preceding label, it looks at the next most likely label of the token and repeats the same check until a label that complies with the constraint is identified. If no such label is found, the label 'O' is assigned to the token. Since the algorithm keeps track of the previous token label, it can ensure that the following token is assigned a most probable label that adheres to the BIO tagging constraints or defaults to the 'O' label.

To develop a different approach that obeys the constraint, we propose an adapted Viterbi algorithm. First, using the training set and MLE, we compute the transition probabilities of all BIO tags occurring, given all previous BIO tags. The log tag transition probabilities are stored in a dictionary of structure: $\{t_{i-1} : \{t_i : P(t_i|t_{i-1})\}\}$, where t_i is the current tag and t_{i-1} is the previous tag. Additionally, BOS and EOS tags are included alongside BIO tag transition probabilities. Instead of calculating the emission probabilities $P(w_i|t_i)$ as required in the original Viterbi algorithm, we utilise the $P(t_i|w_i)$ log probabilities obtained from the most frequent tag or logistic regression model as a replacement. $P(t_i|t_{i-1})$ and $P(t_i|w_i)$ that never occur in the training set are set to $-\infty$. Since the models return the logarithmic probabilities, we perform the summation of the $P(w_i|t_i)$ and $P(t_i|w_i)$ to obtain the emission transition, instead of taking their product as in the original Viterbi algorithm.

The transition probabilities for incorrect tag sequences (I-tag occurring before the respective B-tag) will have a probability of $-\infty$ since they never appear in the training set. Thus, when inferring the tags using the modified Viterbi algorithm of a sample, such tag sequences will have the log probability be $-\infty$ and thus will not be selected as the most probably.

Further, we discuss the drawbacks and benefits of the *bio_tags* and the modified Viterbi algorithms. The *bio_tags* algorithm assigns the I-tag greedily and locally by ignoring the words in the surrounding context. This can result in words getting assigned incorrect BIO tags that have low probabilities to comply with the constraint. The most frequent tag model performance will suffer more than the logistic regression one due to contextual embeddings and more accurate tag probability distribution.

The modified Viterbi algorithm considers the context of the word, specifically the previous word. This allows for label sequences that frequently appear in training set to be assigned more often, whereas label sequences that appear in the dataset rarely have an unlikely chance of getting selected due to their low transitional probabilities. When adding a high $P(t_O|w_i)$ value to the $P(t_O|t_O)$, the algorithm might predict a continuous sequence of 'O' labels, due to the high frequency of the 'OO' label sequence in the training set. In general, miscalculation of one tag can drastically affect the prediction of upcoming tags, thereby resulting in a chain of errors.

3 Error Analysis and Feature Engineering

Further, we examine the classification report of the logistic regression model with the bio_tags constraint applied. we observe that the 'rooms' slot label, which has a comparatively high support of 9, is frequently misclassified, as shown by its low precision (0.22), recall (0.22), and therefore F1 score (0.22). the full report reveals that 14 out of the 21 instances when the true BIO annotation was the 'B-rooms' or 'I-rooms' tag, the model mislabelled it as an 'O' tag. this leads to a high false negative rate, hence negatively affecting the model recall. Additionally, the model suffers from a high false positive rate by predicting the 'B-rooms' or 'I-rooms' tag instead of the correct 'O' tag.

We hypothesise the reason for the model's inability to distinguish between the 'O' and 'rooms' tags to be the absence of specific context. For example, in a validation set sentence 'From the 16th to the 30th, a single room for 2 adults and 1 child.', the model fails to identify 'a single room' as 'rooms' span. This is due to the neighbouring words of the span which provide the context and thus the embeddings appearing in scenarios when the respective span label is 'O'. For example, in a sentence from the training set '22nd until 29th 7 night stay - 2 adults 1 child under 1' the token 'stay' (tag 'O') has similar semantic context to that of the validation 'rooms' span. As a result, the logistic regression model might fail to classify tokens that appear in similar contexts.

We address the problem by incorporating the token dependency tags alongside the original token embeddings. We argue that tokens that might appear in similar semantic contexts but do not share the same BIO tag will have differing dependency tags. Thus, the inclusion of the dependency tags in model features might allow to distinguish between different tag types. From the validation set output, it can be observed that the misclassified 'rooms' spans such as 'a wheelchair friendly room', 'an en suite room', 'double room' and 'a single room' share similar dependency tags, such as – 'det' 'dobj', 'amod' and 'ROOT'. A dependency tag on its own is not sufficient to represent a specific BIO tag, since the same dependency tag might occur in different situations denoting other BIO tags (number and determiner dependency tags). Thus, to ensure the proper context is being passed on to the model, we introduce a window that considers the ± 1 neighbouring token dependencies. To ensure an optimal model training time and limit the amount of noise, we do not experiment with a bigger window size.

In order to incorporate the dependency tags of each word into the model training features, we begin by separately obtaining the counts for all token dependency tags that belong to the 'B' or 'I' tag. The dependency tags are sorted from the least occurring to the most frequently occurring. For each tag, a weight is calculated based on its position in the list:

$$w_i = \frac{i+1}{n},$$

where i - index of the tag in the list, n - number of dependency tags for 'B' or 'I' tags. Further, the dependencies and their respective weights are merged into one dictionary. In the case when 'B' and 'I' weights share the same dependency, the maximum weight is chosen. With such an approach, the weights vary from (0-1], where the most occurring dependency tag is assigned the highest weight and the least frequent dependency tag gets the smallest weight.

After obtaining the spaCy token embedding vector, it is multiplied by the respective dependency weight of the token:

$$\hat{e} = e \times w_i,$$

where e - spaCy token embedding, $\hat{e}$ - modified token embedding that takes into account the token dependency. In the case when the weight for a dependency weight is not present among the calculated weights, we assign $w = 0$. In such cases, we can be sure that the token should

be labelled with the 'O' tag. This process ensures that the resultant token embedding vector contains information about its respective dependency tag relative to its frequency of occurrence in the training set. As a result, undesired tags are eliminated from the input, and the most significant tags are highlighted using weights.

The standard spacy token embedding, e has only the numerical representation of that word in the sentence. Combining e with dependency matrix will result in matrix that holds values that potentially play role in the tag prediction. scenarios where a certain tag has a weight of 0.9 then each feature in e gets amplified with the weight, similarly for tag that has weight 0.1 the amplification is with 0.1 to each values. Hence e values for each token is modified based on it's importance and the role it plays in determining the required output i.e. most important tag sequences will eventually have higher values in e as compared to least influential features.

In order to implement the window of the token, we obtain the previous token embedding $\hat{e_{n-1}}$ and the next token embedding $\hat{e_{n+1}}$. The obtained word embeddings for the window of tokens are concatenated together to form the resultant vector E that can be passed onto the Logistic Regression model.

$$E = [\hat{e_{n-1}}, \hat{e_n}, \hat{e_{n+1}}]$$

By using such an approach, we can expect a relative increase in recall and overall F1 score as the false positive rate might decrease due to the involvement of the dependency in the input. Hence, it reduces the chance of a word that was originally a 'B-rooms' or 'I-rooms' mapping to an 'O' tag.

	precision	recall	F1-score
micro avg	0.42	0.47	0.44
macro avg	0.39	0.43	0.39
weighted avg	0.41	0.47	0.43

Table 1: Slot label results of the standard logistic regression, evaluated on the validation set

	precision	recall	F1-score
micro avg	0.44	0.53	0.48
macro avg	0.41	0.48	0.43
weighted avg	0.46	0.53	0.48

Table 2: Slot label results of my_model, evaluated on the validation set

Upon implementing the my_model, we observe the recall has increased marginally, indicating that the false positive rate has reduced when compared to logistic regression (Table 1, 2). Taking a deep insight into the result, it can be seen that certain tags such as "rooms" (support 9), "adults" (support 9), and "date_period" (support 9) have shown an increase in the F1 score.

Considering the 'rooms' span we notice an increase in precision(0.14), recall (0.22) and F1-score (0.18) for my_model when compared to logistic regression. The model correctly labels some tag spans, such as 'a single room', which has the dependency tags- 'det', 'amod','ROOT'. However, it fails to label a tag span such as 'a wheelchair friendly room', which has the dependencies -'det', 'nmod', 'amod', 'pobj'. Although the correctly predicted dependency tag sequence appears in the training set only once, a similar sequence, such as 'det', 'amod', 'pobj', is present in the training set and has high support. Dependency tags 'ROOT' and 'pobj' have weights of 0.87 and 1, respectively, resulting in similar token embeddings. Whereas the incorrectly predicted sequence never occurs in the training set for the 'rooms' slots and is longer than the context window. The model can be improved by bringing in more training data that contains more combinations of dependency tags for each of the slot tag types to make the model more robust in nature. Further, by increasing the window size, longer slot sequences could be predicted more accurately.

In general, we have shown that my_model has improved the performance with respect to the 'rooms' tag compared to the standard logistic regression. Feature engineering the input data by highlighting needful parameters and bringing in the context for each token has made the model more robust and accurate compared to standard logistic regression.

4 Final Test Reporting and Reflection

When evaluating both models on the test set, we expect to see observations similar to those when evaluated on the validation set. Specifically, my_model should report higher precision, recall and F1 scores than standard logistic regression due to my_model containing information about the dependency tags and context-based predictions, making it more robust compared to the logistic regression. In addition, for some slot labels, we might see a decrease for my_model on the testing set compared to the validation set if the set contains previously unseen dependency sequences of slots or many long slots.

	precision	recall	F1-score
micro avg	0.44	0.44	0.44
macro avg	0.37	0.39	0.37
weighted avg	0.44	0.44	0.43

Table 3: Slot label results of the logistic regression model, evaluated on the test set

	precision	recall	F1-score
micro avg	0.42	0.47	0.44
macro avg	0.39	0.43	0.39
weighted avg	0.41	0.47	0.43

Table 4: Slot label results of the logistic regression model, evaluated on the validation set

	precision	recall	F1-score
micro avg	0.47	0.48	0.47
macro avg	0.46	0.43	0.43
weighted avg	0.52	0.48	0.49

Table 5: Slot label results of my_model, evaluated on the test set

	precision	recall	F1-score
micro avg	0.44	0.53	0.48
macro avg	0.41	0.48	0.43
weighted avg	0.46	0.53	0.48

Table 6: Slot label results of my_model, evaluated on the validation set

Tables 3, 4, 5, 6 indicate that the overall performance of precision, recall and F1 scores of the logistic regression model is relatively lower than my_model's when evaluated both on testing and validation data. As anticipated, there is a slight variation in precision and recall with respect to my_model validation data and test data performance.

Taking a deeper insight into specific slot tags, such as 'rooms', we observe a 0.04 decrease in recall for my_model when evaluated on the testing set compared to the validation set. In the phrase 'I'd like to book two rooms a fortnight today.' my_model fails to label the 'two rooms' with the appropriate 'rooms' slot tags. Upon review, the training data did not contain the corresponding dependency tag sequence ('nummod, dobj') mapping to 'rooms' tags. As a result, the model prediction failed. Additionally, inconsistencies between the training and testing data annotations prohibit the model from learning the relevant dependency tag patterns. For example, the token sequence '2 rooms' in the training set has varying annotations. Specifically, the sequence is classified as 'B-rooms I-rooms' on some occurrences and as 'B-rooms O' on others.

Although the majority of the slot tags show an improvement in both precision and recall when evaluating my_model on the test set, some date and time-related slot tags, such as 'date', 'date_period' and 'time_period' show a major decrease in both precision and recall. My_model confuses the labels with one another due to their similar dependency tag sequence structure.

Scenarios like this can be improved by bringing in training data that contains more combinations of dependency tags for each of the slot labels and having a consensus of BIO tags for similar slots to make the model more robust in nature. Another improvement is to increase the window length by increasing from ± 1 to ± 2, which might bring in more context of a phrase with respect to the token into the picture.