ANLP Assignment 2 2024

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

Use this template file for your solutions. Please start each question on a new page (as we have done here). Do not remove the pagebreaks. Different questions may be marked by different markers so your answers must be self-contained.

Words	Truth Set	Predicted Set
bruggeman	B-person_name	O
shaida	B-person_name	O
delana	B-person_name	O
jon	O	O
jacuzzi	O	O
mural	B-person_name	O

(a) Most frequent Tag model

Words	Truth Set	Predicted Set
bruggeman	B-person_name	B-date_to
shaida	B-person_name	B-person_name
dalana	B-person_name	B-person_name
jon	O	B-person_name
jacuzzi	O	B-person_name
murad	B-person_name	O

(b) Logistic Regression model

Table 1: Labels and predicted tags for each word for both models

1 Logistic regression tagging with word embeddings

In comparing the two models, we see that all three types of averages (micro-average, macro-average, and weighted average) for all metrics reveal better scores for the Logistic Regression Model, almost doubling the scores from the most frequent tag model. A significant improvement across these three averages indicates that the Logistic Regression Model performs better. Take as an example the weighted average for precision, recall, and F1-score; the most frequent tag model yields the following values: 0.32, 0.22, 0.24, respectively, while the Logistic Regression Model gives 0.53, 0.49, 0.50.

Delving the B-person_name tag, we provide examples in Table **??**. We observe that the frequent tag model performs poorly on words with the B-person_name tag, assigning the O tag to all of them, as depicted in Table **??**. This reveals one of the models' weaknesses, which we discuss further below. On the other hand, the Logistic Regression Model successfully identifies the correct tag for some tokens in Table **??**, leading to more accurate recognition of names, even though it occasionally introduces false positives.

Analyzing the most frequent tag model, we observe that it struggles to assign correct tags to unseen words (words that appear in the validation set but were absent in the training set). This issue arises because the probability of tags given these unseen words is zero, as the word-tag pair does not exist in the training set. As a result, the model defaults to assigning these words the O tag. This behavior is explained by the maximum likelihood estimate (MLE) formula:

$$P(t \mid w) = \frac{\text{count}(t, w)}{\text{count}(w)}$$

Where $\text{count}(t, w)$ is the number of occurrences in the training data set such that a token w appears with tag t. This value is zero for unseen words, regardless of the tag; meaning that $P(t|w)$, the probability of assigning tag t to a word w, will also be zero. As a result, the model assigns the O tag to these words.

On the other hand, the logistic regression model makes its predictions based on word embeddings, as opposed to the words themselves. Since $P(t \mid w)$ is now a logistic regression model, we have that $P(t \mid w) \in (0, 1)$ which is never zero, avoiding the previous problem.

Additionally, embeddings provide more context for name entity recognition (NER) systems by capturing word meanings and relationships. This allows the model to link new or unseen words to similar ones from the training set, improving its ability to correctly assign tags based on probabilities calculated for each embedding. This approach reduces the number of words being misclassified with the O tag, as seen in Table **??**. Overall, we can see that the Logistic Regression Model performs better than the most frequent tag Model.

2 BIO tagging for slot labeling

The predict_bio_tags() function generates all possible tags for each token based on word embeddings, sorted by their probability. It then retrieves the most probable tag and checks if it meets the BIO tagging constraint. Specifically, if the tag starts with the prefix I-, the function compares its type with the type of the previous tag. If they don't match, the function moves to the next most probable tag. Once a tag satisfies the constraint, it is then assigned to the token. In other words, since BIO tags begin with I- or B-, we are ensuring that every I- follows a B- or I- tag of the same type (BIO constraint). This procedure is repeated for each token in the input sentence.

Word	predict_independent_tags()	predict_bio_tags()
5	B-date_period	B-date_period
nights	I-date_period	I-date_period
from	O	O
15th	B-date_from	B-date_from
of	I-date_to	I-date_from
October	B-date_from	B-date_from
?	O	O

Table 2: BIO Predictions using predict_independent_tags() and predict_bio_tags()

Observing Table **??**, the model using predict_independent_tags() assigns the tag I-date_to to the token of. However, I-date_to does not align with the BIO constraint (the tag type date_to does not match with the tag type date_from of token 15th), so predict_bio_tags() replaces I-date_to with I-date_from, which is the most probable tag that satisfies the constraint.

As a suggested approach, the Viterbi algorithm is well-suited because it uses dynamic programming method to find the most likely sequence of tags in a Hidden Markov Model (HMM) given an observation sequence. In a HMM, the probability of each tag depends only on the previous tag (Markov Assumption), allowing us to enforce BIO constraints by defining valid and invalid transitions. Specifically, if the current tag (state) begins with I-, the previous tag (state) must begin with B- or I- of the same type; otherwise, the transition probability is set to zero. Additionally, emission probabilities are given by $b_s(o_i) = \frac{P(t|o_i)}{P(t_i)}$, where o_i is the i^{th} token of the sentence, and t denotes the tag of state s; $P(t|o_i)$ is given by our model and $P(t_i)$ is computed through MLE. At each step, the algorithm uses both transition and emission probabilities to compute the most likely path with a correct BIO tagging format by skipping invalid transitions.

Before applying the Viterbi Algorithm, the transition probabilities need to be computed. In detail, the probability of valid transitions are computed through Maximum Likelihood estimations while invalid transitions are set to 0. The computational complexity is $\mathcal{O}(S^2 \times T)$, where S is the number of tags and T is the sentence length. The complexity comes from calculating each possible tag transition (from each previous tag to each current tag) at each position in the sentence. In contrast, the complexity of the provided algorithm is $\mathcal{O}(S \times T)$, in the worst scenario, as it checks all the possible tags for each token. However, the typical complexity is $\mathcal{O}(T)$, since the most probable tag often complies with the BIO taggings rule.

Overall, the provided algorithm is more efficient and requires less memory, making it suitable for longer sequences. Unlike Viterbi algorithm, it doesn't store backpointers, which Viterbi uses to trace back from the last token to determine the optimal tag sequence

Finally, the Viterbi algorithm provides more accurate results by considering dependencies between tags across the sequence. In Table **??**, Viberti would likely help the model recognize October as part of the span 15th of October and correctly assign it the tag I-date_from.

3 Error Analysis and Feature Engineering

We identified that the current model frequently misclassifies the span label date_from. We examined this issue closely, presenting specific examples along with reasoning as to why these errors might occur and suggesting potential ways to address them.

Considering this, the model sometimes fails to recognize tokens, which should be assigned to date-related tag labels, such as date, date_from, and date_to. In the following example "A double room for tomorrow for 8 days", the model does not tag tomorrow as date-related; instead, it incorrectly marks it as O (outside tag), as shown in Table **??** below. This issue arises because the embeddings may not provide enough information to identify the type of entity each word represents, as indicated by the Entity Type column in Table **??**.

Word	**Entity Type**	**BIO Annotations**	**Original BIO Predictions**	**Updated BIO Predictions**
a	-	B-rooms	B-rooms	B-rooms
double	-	I-rooms	I-rooms	I-rooms
room	-	I-rooms	I-rooms	I-rooms
for	-	O	O	O
tomorrow	DATE	B-date_from	O	B-date
for	-	O	O	O
8	DATE	B-date_period	B-date_period	B-date_period
days	DATE	I-date_period	I-date_period	I-date_period

Table 3: Comparision between BIO Predictions of the Original and Updated Logistic Regression Model incorporating the linguistic feature entity type of each word

To reduce the problem, we decided to add the linguistic feature of ent_type_ to the embeddings. We believe this approach will be effective because token.ent_type_ frequently recognizes common entity types such as dates. This provides the model with clearer indications that certain words, such as tomorrow should be assigned with a date-related tag.

The results of the updated model in Table **??** show that tomorrow is now assigned to a date-related tag, B-date.

However, the model still struggles to accurately distinguish between similar span tags, such as date, date_from, and date_to. This challenge occurs because each tag has a specific contextual meaning that often depends on surrounding words. For instance, the tag date_from usually relies on words such as from or starting, while date_to might depend on words to or until.

Word	**Token Head**	**BIO Annotation**	**Original BIO Predictions**	**Updated BIO Predictions**
Two	nights	B-date_period	B-date_period	B-date_period
nights	nights	I-date_period	I-date_period	I-date_period
from	nights	O	O	O
Friday	29th	B-date_from	B-date_from	B-date_from
29th	from	I-date_from	B-date_to	I-date_from

Table 4: Token's Head, BIO Annotations, and BIO Predictions of the Original and Updated version of the Logistic Regression Model.

Analyzing the sentence "Two nights from Friday 29th", as shown in Table **??**, the token 29th should be labeled as date_from, based on the context provided by the word from. But the original model incorrectly assigns the tag type date_to to 29th, interpreting it as the end of the date range rather than part of a starting date. This misclassification occurs because the current embeddings do not capture the relationship between 29th and its neighboring word from. The existing embeddings represent each token individually and may lack information about this kind of contextual links between tokens.

To improve the model's performance and address this tagging issue, we further added an additional linguistic feature, token.head, to enrich even more the embeddings' information. By concatenating token.head.vector to the previous modified embeddings, this feature provides more context about how each word in a sentence connects to others, especially highlighting which words act as heads that influence the meaning of neighbouring words.

In Table **??**, token head offers meaningful information because dates are often surrounded by words such as from, to, or until that indicate a time range. In the following sentence ”Two nights from Friday 29th”, the token head feature leads the model to recognize that 29th is linked to the token from, as illustrated in Figure **??**. With this addition, the model can correctly assign the token 29th to the tag type date_from rather than date_to, as reflected in the updated predictions in Table **??**.

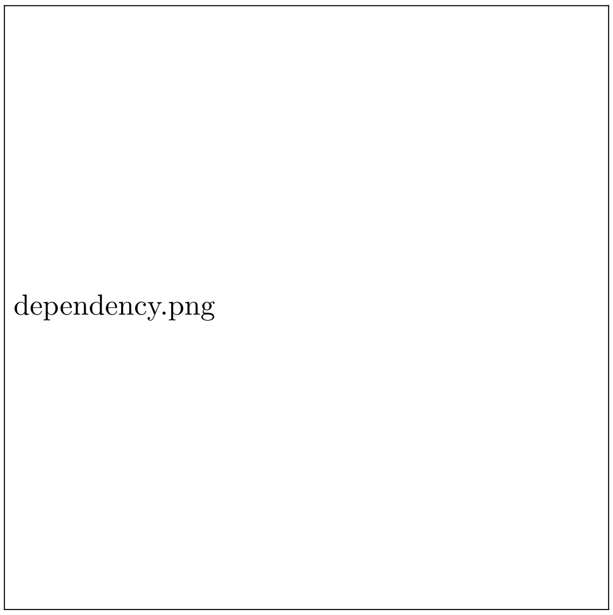

Figure 1: Visualization of dependency parse, showing the head relationships for each token.

In summary, the updated Logistic Regression Model, enhanced with embeddings that incorporate additional linguistic features, demonstrates an improvement over the original. The average metrics across all slot labels show higher values, as shown in Table **??**, indicating the benefits of including features, such as head dependencies and entity types. Remarkably, we noticed that the three metrics (Precision, Recall, and F1-score) of the slot label date_from are significantly better increasing from 0.44, 0.55, 0.49 in the original model to 0.62, 0.65, 0.63 of the updated model, respectively.

Going further, we observe that the modifications made to our model yield promising results across most slot labels. For example, in the sentence From the 16th to the 30th, a single room for 2 adults and 1 child, the updated model correctly assigns the slot label rooms to the tokens a, single, and room, identifying the word room as their syntactic head. In contrast, the original model incorrectly labeled these tokens as O. By incorporating head dependency, the model is better equipped to recognize when tokens belong to the same span, reducing misslabeling errors. This improvement makes the model to perform better for similar cases, such as 2 adults or 8 people, labeling those correctly through their syntactic relationship (the head of 2 is adults and

	Original Model			**Updated Model**		
Metric	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
Micro Avg	0.47	0.47	0.47	0.56	0.54	0.55
Macro Avg	0.40	0.41	0.40	0.45	0.45	0.44
Weighted Avg	0.46	0.47	0.46	0.55	0.54	0.53

Table 5: Comparison between the averages of Precision, Recall, and F1 Score for Original and Updated Models

people is the head of 8).

4 Final Test Reporting and Reflection

Starting with the comparison of the original Logistic Regression model on both the validation and test sets, we observed from the classification report that the model performs better on the validation set, as shown by the average metrics, which provide an overall assessment of model performance. For example, the micro averages for Precision, Recall, and F1-score are 0.37, 0.39, and 0.38 on the test set, compared to 0.47 for all metrics on the validation set. This pattern is similar in our model, with micro average metrics of 0.43, 0.44, and 0.43 for precision, recall, and F1-score on the test set, while the validation set scores were higher at 0.56, 0.54, and 0.55.

Since we didn't conduct hyperparameter optimization on the original Logistic Regression Model using the validation set, we cannot explain with certainty the performance discrepancy between the validation and test sets. Given that the training, validation, and test sets were randomly sampled from the same source, we would expect similar performance between validation and test sets. However, if the validation set happens to be more closely aligned with the training data, it may yield higher performance scores. On the other hand, the test set could contain unique or less frequent patterns, leading to lower performance metrics.

Unlike the original Logistic Regression model, our updated model was refined based on results and examples from the validation set with additional linguistic features to help it recognize patterns in the validation data and assign correct tags to tokens. However, because these patterns are less frequent in the test set, the model's performance is lower on this set than on the validation set.

To be specific, our model is able to recognize the pattern "from [DATE]". For example, in the phrase from 15th of October, the model correctly assigns the tag sequence B-date_from, I-date_from, I-date_from to the tokens 15th, of, and October, respectively. However, this pattern is less common in the test set, appearing only twice compared to seven times in the validation set, even though the overall support for date_from tags is similar. This difference in pattern frequency leads the model to perform worse on the test set.

Overall, the modified model with extra linguistic features performs better than the original logistic regression model across the classification metrics on both the validation and test sets. This is mainly because the added features give the token embeddings richer context from surrounding words, helping the model make more accurate tag predictions and reducing misclassifications. As a result, the modified model shows improved performance on both the validation and test sets compared to the logistic regression model.