Improving Slot Tagging Accuracy with Bidirectional LSTMs and Class Imbalance Techniques

Chelsey Rush
UC Santa Cruz
NLP243 Homework 2
chrush@ucsc.edu

Abstract

Slot-tagging is a key application of natural language processing. Similar to named entity recognition, part-of-speech tagging, and chunking, slot-tagging is a **sequence labeling task**. These tasks involve processing sequences of human language and assigning a label to each element in that sequence. Just as other tasks, slot-tagging presents its own unique challenges, especially when working with an *imbalanced dataset*. This paper proposes a recurrent neural network with a long short-term memory (LSTM) architecture enhanced with bidirectionality and class weighting, designed to reduce the negative effect of the imbalanced labels on the model's predictive capacity. Furthermore, at the dataset level, this paper investigates the effect of oversampling techniques in improving the model's accuracy and ability to generalize. Experimental results suggest that a **bidirectional LSTM with inverse class frequency weighting and oversampling** achieved notable improvements for rare classes, supporting its performance in imbalanced slot-tagging tasks.

1 Introduction

Slot tagging is one of the key tasks in natural language processing and understanding and aims to i) disambiguate the intention of an utterance and ii) label relevant sequences of the utterance with semantic tags. This task requires that each token of a user utterance be labeled with relevant slot tags. These utterances were given by a human user to a virtual assistant and mostly focused on film-related questions.

Because each token must be categorized and labeled with a single slot label, this task is a supervised, multi-class classification task. By providing reliable, accurate slot tags to these user utterances, the virtual assistant system will be able to efficiently and correctly retrieve relevant information from a backend knowledge graph and supply the user with a satisfactory response.

2 The Dataset

The dataset for this task consisted of 2297 pre-annotated utterances with slot tags in a common IOB (inside, outside, beginning) format, wherein the token's location with the entity is flagged with the I, O, or B label, followed by the category of the slot. All non-relevant slots are labeled with O, indicating that the given token is not relevant to the user's intention within the utterance. In addition, the text has been preprocessed into lowercase, with individual tokens already separated for each utterance. The slot tags appear in the *IOB_category* format, an example of which is shown below.

```
ID: 1726
UTTERANCE: search the movie in
    july
IOB Slot Tags: O O O B_movie
    I_movie
```

As *O* indicates that the given word is not considered a meaningful slot, it is unsurprising that this label has the highest frequency in the data. This label occurs 10,428 times in the training data.

The frequency of the other tags is shown in Figure 1 and Table 1. **The training data is quite imbalanced** regarding category, with the *movie* category occurring over four times as often as the next most frequent category. Additionally, **some categories were more likely to contain longer sequences of tokens**, i.e. I (inside) slot tags, such as the *cast*, *director*, *movie*, *person*, and *producer* categories. On the other hand, categories such as *char*, *country*, *genre*, *language*, *mpaa_rating*, and *subject* were less likely to contain I slot tags, meaning these slots were often single tokens.

3 Models

3.1 The Base Model

- **Model Type:** The base model for this task is a unidirectional Long Short-Term Memory

Category	B	I
location	2	0
release_year	5	3
char	14	5
genre	70	5
subject	95	33
cast	105	104
language	117	17
mpaa_rating	141	12
country	153	12
producer	164	114
director	184	167
person	193	174
movie	1009	1133
O: 10428		

Table 1: A table displaying the category distributions of class labels within the training data grouped by category and IOB tag.

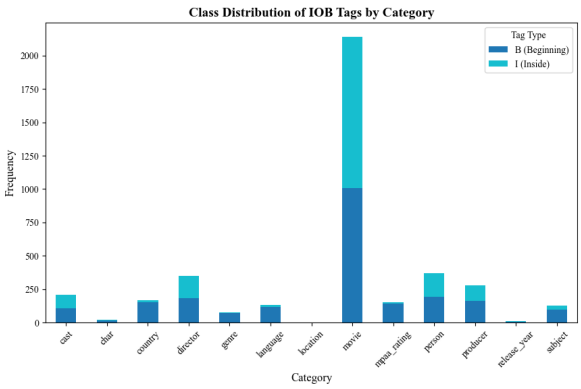

Figure 1: A stacked bar chart displaying the category distributions of IOB tags within the training data. Dark blue indicates a B (beginning) tag, while light blue indicates an I (inside) tag for each category. Distribution of the *O* tag has been left from this plot to maintain the visibility of the other labels.

(LSTM), which outputs a prediction for each token in the input sequence and consists of an embedding layer, one LSTM layer, and a fully connected output layer.

- **Embedding:** The base model for this task used a custom embedding. A custom, trainable embedding layer that maps token IDs from the vocabulary to dense, 100-dimensional vectors that capture the token's semantic content. This approach is advantageous because the embeddings are learned and optimized during training: as the model updates its parameters based on the computed loss, the embeddings are also updated. The intention is that the model will become better able to learn the relationship between the tokens' features and the expected slot tags, and tailor the embeddings specifically to the task data.

- **Training Process:** The training process for the base model used a **batch-learning** approach with a batch size of 128. In addition, **cross-entropy loss** was selected for this task because this function is well-suited for multi-class sequence labeling tasks, especially those with categorical outputs. After the loss calculation, **the AdamW optimizer** updated the model parameters. The AdamW optimization has been shown to result in higher accuracy for deep learning tasks Zhang (2018) than vanilla Adam and implements weight decay independent of the gradient calculation. Several techniques were applied in order to prevent overfitting: **early stopping** when the validation loss ceased decreasing for 3 or more epochs and a **0.5 dropout rate** during training.

- **Tuneable Hyperparameters:** As an LSTM, this model included several tuneable hyperparameters: embedding dimension size, hidden dimension size, batch size, learning rate, number of epochs, the loss function, and the optimizer. The optimized values for each of these is as follows: embedding dimension of 800, hidden dimension of 128, a batch size of 128, a learning rate of 0.005, and number of epochs set to 20.

- **Evaluation Metrics:** As a sequence labeling task, this problem required a specialized evaluation metric. Throughout this report, any accuracy or F1 score reporting is made using the F1 score metric from the *seqeval* Python framework for sequence labeling evaluation Nakayama (2018).

4 Experiments

The dataset was split so that 90% of the data was used to train the model, and the remaining 10% was reserved for model validation and hyperparameter tuning. Hyperparameter optimization was carried out using a grid search. Models' performance was

evaluated using the class distribution of the output predictions, the F1 score of the model on the validation and test sets, and by analyzing the misclassifications during the validation phase.

It is also important to note that the F1 score for this task was computed using Nakayama (2018)'s IOB2 scheme, a Python framework specifically engineered for sequence labeling evaluation.

- **Experiment 1: Implementing a Bidirectional LSTM to Capture Context-Dependent Relationships.**

  Huang et al. (2015) found that implementing a BI-LSTM outperformed LSTM and found that bidirectional LSTMs were less reliant on engineered features and performed better on sequential data. While a unidirectional LSTM can only use the tokens it has already encountered in a sequence, a bidirectional LSTM can use the context from past *and* future tokens to predict the tag for any given word. Because slot tags are deeply relational and a single slot can span across many tokens, it is hypothesized that a bidirectional model might capture these dependencies and better generalize them to unseen data.

  Initial explorations into the predictions from the Base Model revealed that the unidirectional LSTM predicted 120 slot tags with the *I* slot label that did not have preceding *B* tags, which should be impossible considering no token should be tagged *inside* without first tagging the *beginning* of the entity. It is hypothesized that incorporating bidirectionality into the model will lessen the number of I tags without preceding B tags.

- **Experiment 2: Class Weighting and Label Smoothing to Address Class Imbalance**.

  As shown in 2, the categories represented in the training data are unbalanced, which could lead to the model being unable to predict underrepresented labels in the data accurately. In order to illustrate this, the class distributions of the predicted slot tags from the Base Model are shown in Figure 2. The predicted label distributions mimic that of the training data quite closely, but fails to accurately predict any instances of *char*, *location*, or *release year* in the validation set.

  To address this imbalance, different class weighting and data augmentation techniques within the model's loss function; this assigns higher weights to less frequent tags, and oversamples rarer tags in hopes that the model becomes more sensitive to them Hasanin et al. (2019). The class weighting introduced was the inverse frequency class weighting, where more frequent tags were assigned less weight than rarer tags. Furthermore, the effect of dataset-level techniques such as over and undersampling on accuracy and F1 score were observed.

5 Results

5.1 The Base Model

The base model achieved a decent accuracy for the validation and test sets: an F1 score of 0.772 on the validation set and an F1 score of 0.675 on unseen data. The training loss decreased from 1.44 to 0.023 throughout the 15 epochs, and the validation loss decreased from 0.696 to 0.243. The class distribution for this model's predictions on unseen data is shown in Figure 2. Although this model generated predictions that mimicked the class distribution from the data, it often misclassified underrepresented classes such as *B_movie* or *I_movie* and often classified items as *Inside* slots without a preceding *Beginning* slot.

5.2 The Bidirectional LSTM.

Implementing a bidirectional LSTM instead of a unidirectional LSTM increased the validation accuracy from 0.75 to 0.83 after 20 epochs. Furthermore, the model's accuracy when predicting unseen data rose from 0.67 to o.73.

First, to observe how changing the model's architecture changed the model's predictions, the class distribution of the models' predictions were visualized and shown in Figure 2. Unsurprisingly, the Base Model's class distribution resembles the distribution within the training data, with items such as *location* and *release_year* having a very small number of output predictions, *char* having none at all. On the other hand, the BiLSTM was able to predict some *char* labels, but still struggled predicting *location* and *release_year* labels. Furthermore, the distribution of B and I *director*, *person*, and *producer* labels became more representative of the training data, becoming about 50% I and 50% B slot tags – which is logical considering names are usually two tokens.

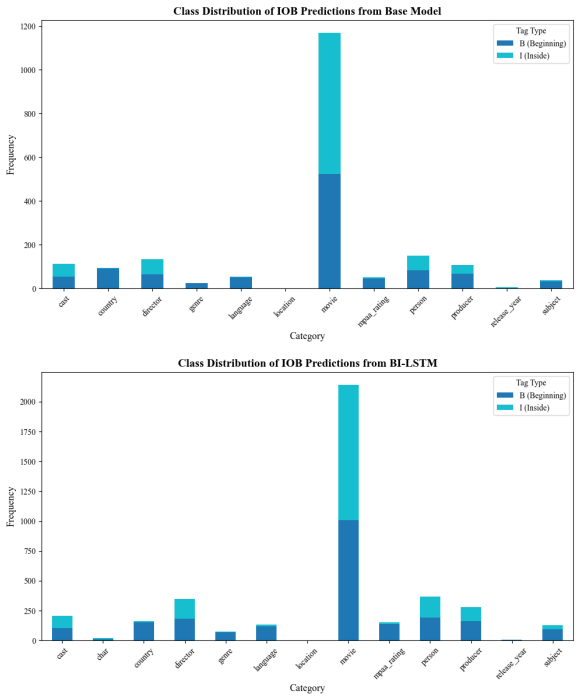

Figure 2: Class distributions of IOB predictions from the unidirectional Base Model and bidirectional LSTM.

The macro-average F1 scores for the unidirectional and bidirectional LSTM and the number of I tags without preceding B tags are shown in Table 2. **Implementing a bidirectional LSTM increased the model's macro-average F1 score by 4%**, increasing the model's ability to predict underrepresented classes in the dataset.

Model	Macro-Average F1	# of I Initial Tags
UniLSTM	0.75	248
BiLSTM	0.79	120

Table 2: Macro-average F1 scores and number of predicted I slot tags without preceding B tags for both the Unidirectional LSTM and the Bidirectional LSTM.

Furthermore, the number of *Inside* tag predictions that did not have a preceding Beginning tag was reduced by nearly half, from 248 to 120. This suggests that **implementing bidirectionality within the model allowed the model to retain insightful information regarding how order related to the labeling of the slots given the sequences**.

In addition, confusion matrices for the misclassified items are shown in Figure 3. Implementing a BiLSTM reduced the number of *O* slots incorrectly labeled as *I_movie* from 365 to 193 and the number of B_movie slots incorrectly labeled as *O*

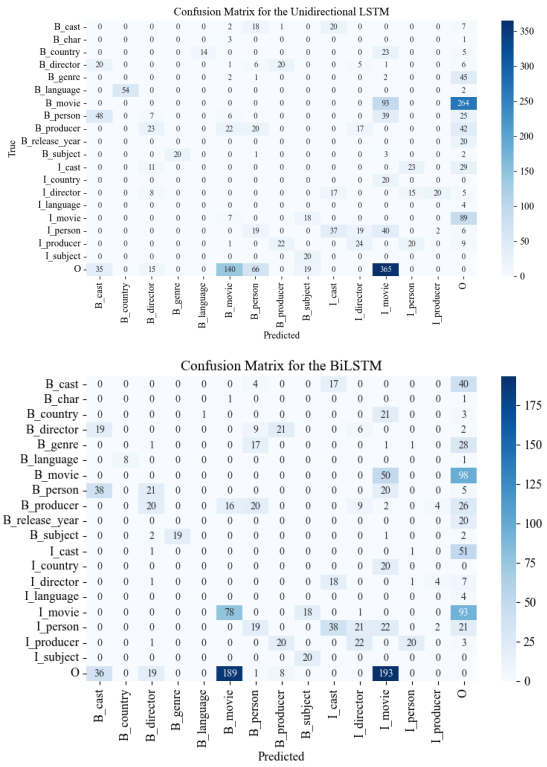

Figure 3: Confusion matrices of misclassified items for the Unidirectional and Bidirectional LSTMs.

from 264 to 98. In addition, the model greatly improved F1 scores for underrepresented labels in the data, reducing the number of items misclassified as B_country from 54 to 0, and the number of items misclassified as B_language from 14 to 1.

However, although implementing a bidirectional architecture generally reduced the number of misclassifications across most of the labels, it seems that the number of items misclassified as *B_movie* and *B_director* increased. **This suggests that although the bidirectional LSTM architecture improves model accuracy by augmenting its contextual awareness, the model may become less sensitive to distinctions delineating between some classes.**

5.3 Class Imbalance Methods.

The macro-average F1 score at the final epoch of the training was reported for each model and is shown in Table 3. Oversampling and inverse frequency class weighting both resulted in improved macro-average and weighted-average F1 scores compared to the base model or a bidirectional LSTM.

The model's macro-average F1 score increased from 0.75 to 0.80 when counter-imbalance tech-

niques were applied. Furthermore, the model's accuracy on unseen data increased from 0.73 to 0.77 on unseen data.

Model	Macro-Average F1
Base Model	0.75
BiLSTM	0.75
OS10-BiLSTM	0.78
OS100-BiLSTM	0.79
OS200-BiLSTM	**0.80**
OS300-BiLSTM	0.79
OS10-BiLSTM	0.78
IF-BiLSTM	0.79
IF-OS100-BiLSTM	0.79
IF-OS200-BiLSTM	**0.80**
IF-OS300-BiLSTM	0.79

Table 3: Macro-average F1 scores for each of the methods for addressing class imbalance.

Introducing either oversampling or inverse frequency class weighting improved the macro-average F1 score of the model by at least 3%. **Oversampling rare labels, even by only 10 occurrences, increased the accuracy of the model's predictions.** Oversampling rare tags by 200 utterances and combining this oversampling with inverse frequency class weights tied for resulting in the highest accuracy of 0.80 macro-average F1. However, it is important to note that combining both techniques often results in similar accuracy scores. This might suggest that *oversampling alone* may be the preferred method for increasing the model's ability to predict rare tags without overcomplicating the training process with extra calculations.

In order to get a closer look at exactly how much these methods of addressing class imbalance were impacting the models' ability to predict rarer terms, the confusion matrices of the misclassified items for the base model, the bidirectional LSTM without imbalance measures, and the two best-performing models with the class imbalance measures were evaluated, these are shown in Appendix A.

The most misclassified labels for each model were the B_movie and I_movie tags. The BiLSTM incorrectly predicted 310 *O* tags as *I_movie*, and 303 *B_movie* tags as *O*. This model also struggled to label inner slots of the *movie* category, incorrectly labeling 140 *O* labels as *B_movie*.

The confusion matrices of misclassifications for both models further evidence that implementing oversampling alone produces similar accuracy improvement as implementing oversampling and inverse frequency class weighting. Implementing oversampling greatly reduced the number of misclassifications for each of the most commonly misclassified labels, the number of *O* tags incorrectly predicted as *I_movie* was reduced from 310 to 190, and the number of *B_movie* tags misclassified as *O* was reduced from 303 to 119. Interestingly, the number of *O* labels mislabeled as *B_movie* increased from 142 to 159.

These results suggest that addressing class imbalances at the model and dataset level can increase the model's ability to predict rarer classes but may do so at the expense of predicting more common classes with reduced accuracy.

6 Conclusion

This paper explored the impact of bidirectional LSTMs, class weighting, and oversampling techniques on a slot-tagging task with an imbalanced dataset. The results suggest that incorporating bidirectionality significantly improved the model's ability to capture context information and relationship dependencies, which led to improved performance in predicting underrepresented classes. In addition, strategies for addressing class imbalance, such as inverse frequency class weighting and oversampling further improved the accuracy of the model both in the validation and test phases. However, both experiments suggest that increased accuracy for underrepresented classes came with a cost of reduced accuracy for more common labels. Further work might explore methods for increasing accuracy for rare labels without sacrificing the model's predictive power for common labels.

7 References

References

Tawfiq Hasanin, Taghi M Khoshgoftaar, Joffrey L Leevy, and Naeem Seliya. 2019. Examining characteristics of predictive models with imbalanced big data. *Journal of Big Data*, 6:1–21.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Hiroki Nakayama. 2018. seqeval: A python framework

for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Zijun Zhang. 2018. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, pages 1–2. Ieee.

A Appendix A: Confusion Matrices for Class Imbalance Methods

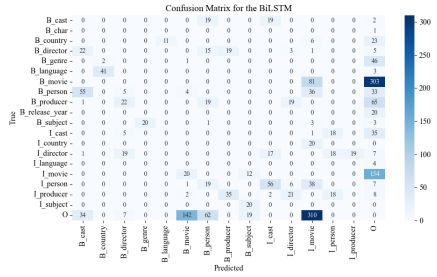

Figure 4: Confusion matrix for the BiLSTM.

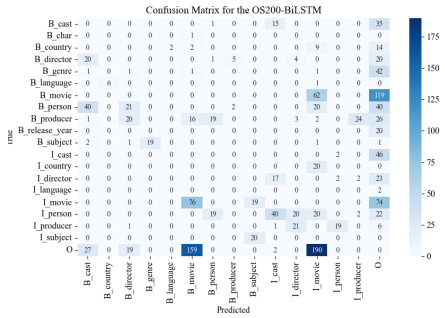

Figure 5: Confusion matrix for the OS200-BiLSTM.

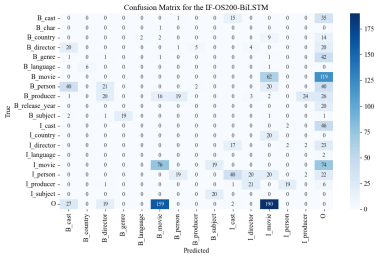

Figure 6: Confusion matrix for the IF-OS200-BiLSTM.