Model Adjustments for Improving Novel Slot Detection

Victoria Graf
Princeton University
vgraf@princeton.edu

Ashley Teng
Princeton University
ateng@princeton.edu

Brendan Wang
Princeton University
byw2@princeton.edu

Abstract

Novel Slot Detection (NSD) is a modified version of slot filling introduced by (Wu et al., 2021) that allows slot types not in the predefined set to be identified at test time. Wu et al. (2021) propose a basic framework for a novel slot detection model involving an embedder, encoder, and classifier used for this new tagging task. The original model contained a BERT embedder and BiLSTM encoder. In this work, we propose three modifications to this framework to study model behavior: (1) adjustments to dropout and patience, (2) replacing the BERT embedder with ELMo and the OpenAI transformer, and (3) replacing the BiLSTM encoder with GRU and self-attention encoders. Using F1 scores and ROSE metrics, we find that the hyperparameter changes did not improve the performance, and the ELMo and OpenAI models achieve lower performance than the BERT model. We observe that GRU and self-attention models can outperform the BiLSTM model on some datasets, and that self-attention encoders are a promising future direction for improving the NSD task.

1 Introduction

Slot filling is a sequence tagging task important to query-based interfaces such as personal assistants. Specifically, tokens in a query are tagged with a slot type such as "playlist" or "movie" for use in downstream tasks. However, typical slot-filling models are limited to predefined slot types used in training, so these models are unable to handle new slot types and may fail to identify a novel slot or may tag it as an incorrect type. This restriction is particularly limiting in light of applications of slot detection in personal assistants such as Amazon Alexa, Microsoft Cortana, or Apple Siri where users may frequently request functionality not yet implemented or available in the interface, a limitation also known as out-of-domain intent detection (OOD) (Wu et al., 2021).

A paper by Wu et al. (2021) suggests a modification of the slot-filling task called Novel Slot Detection (NSD) which introduces, in addition to slot-type and null tags, a novel-slot tag for slots that do not fit one of the previously defined types (Wu et al., 2021). The novel-slot tag aims to address OOD limitations on the token level by identifying novel slots (that is, slots not in the predefined set of tags).

1.1 Problem

In the original slot-filling task, a sentence $\vec{s} = \{w_1, \ldots, w_n\}$ with n tokens is assigned a corresponding predicted sequence of tags $\vec{y} = \{y_1, \ldots, y_n\}$ where y_i is a BIO tag. BIO tags take one of three forms: B-slotType (the beginning of a slot of type slotType), I-slotType (an intermediate word in a slot of type slotType), and O (null tag − not in a slot). Each slot type comes from a predefined set of slot types to create a set of tags $\mathbf{y}$ (such that $y_i \in \mathbf{y}$) which is fixed before training. This task, using only the BIO tags with predefined slot types, is called in-domain (IND) slot filling. In the NSD task, a fourth tag type NS is introduced for novel slots. Like the O tag, NS tags are not split into beginning (B) and intermediate (I) tokens (Wu et al., 2021).

2 Related Work

Current work on slot filling focuses on the similarities this task has to intent detection (Weld et al., 2021). While slot filling focuses on tagging which tokens in a query carry information necessary to execute the user's intent, intent detection is a supervised classification task over the whole query. Several authors have published joint intent-detection and slot-filling BERT models that show significant

improvement over pure slot-filling benchmarks to reach slot-filling F1 scores of 88.6% on ATIS and 92.8% on Snips data with 10 and 20 epochs respectively (Han et al., 2022). Additional studies have shown F1 scores of up to 97.0% on Snips with 30 epochs, which dropped to 95.8% following ablation analysis removing the joint intent-detection task (Chen et al., 2019).

One major issue with the current slot filling task is the limited predetermined set of slot types. Notably, OOD limitations in intent detection have been addressed by work comparing OOD and in-domain intent samples using various detection algorithms such as Maximum Softmax Probability and Gaussian Discriminant Analysis to identify out-of-domain intents (Hendrycks and Gimpel, 2016; Xu et al., 2020). However, similar issues in slot filling have not received the same treatment. Wu et al. (2021) adds a new dimension to slot-filling tasks by introducing the novel-slot tag to form the novel slot detection task. Specifically, they established a model framework, datasets, and evaluation metrics for their proposed NSD task. In their framework, Wu et al. (2021) was able to establish basic trends in NSD vs. IND task performance, namely that NSD performance increases with proportion of unknown slot types while IND performance decreases (Wu et al., 2021). Additionally, they were able to establish the usefulness of their proposed new ROSE metric by comparison to the standard slot-filling metric of span F1 (see Section 5) (Wu et al., 2021).

Since novel slot detection is a new task, alternative frameworks have not been explored and evaluated. Only BERT-based models have been considered for the NSD task, and there are no benchmarks that use autoregressive models. Autoregressive models (such as GPT) only look at previous context to predict future tokens while autoencoder models (such as BERT) use context from both directions (Yang et al., 2019). While autoencoder models are likely to perform better in the NSD task, no empirical results have been shown to support this claim. To explore the importance of framework choice on NSD performance, we consider the following modifications to the framework and model introduced by Wu et al. (2021) for NSD: (1) altering model hyperparameters to determine the best choice of hyperparameters (2) replacing the BERT token embeddings with ELMo (bidirectional autoregressive) (Peters et al., 2018) and OpenAI transformer (Radford et al., 2018) embeddings to ensure that BERT is an empirically sensible choice for this framework (3) replacing the LSTM seq2seq encoder with a GRU and other self-attention encoders.

Ground Truth	play O	hello B-song	by O	**lionel** **B-artist**	**richie** **I-artist**
Replace	play O	hello B-song	by O	**lionel** **O**	**richie** **O**
Mask	play O	hello B-song	by O	**MASK** **O**	**MASK** **O**
Remove	- -	- -	- -	- -	- -

Table 1: Processing of an example sentence using the three strategies Remove, Mask, and Replace where "artist" is considered a novel slot.

3 Data

The original Snips dataset (Coucke et al., 2018) is a collection of queries distributed over several user intents such as weather or music. The dataset contains 13085 train, 700 validation, and 700 test utterances. For our experiments, we used processed Snips datasets modified for the NSD task (Wu et al., 2021). Wu et al. (2021) constructed these datasets by randomly selecting a portion of the slot types to be "unknown slots" at test time. The Snips 5%, 15%, and 30% datasets denote datasets with 5%, 15%, and 30% of the slot types labeled as novel slots.

Following construction of the datasets, sentences containing both novel and in-domain slots need to be properly handled. Thus, the training data were processed using three strategies: Replace, Mask, and Remove. In Replace, the labels of novel slots are replaced with "O". Mask similarly modifies the labels of novel slots, but their corresponding values are also replaced with a special "MASK" value. Finally, Remove eliminates all sentences that contain novel slots. We provide an example of processing on one sample in Table 1. Wu et al. (2021) notes that Remove performed the best empirically and also best simulates real world applications where training data does not contain novel slots (Wu et al., 2021). Thus, this paper only considers data processed with the Remove strategy.

4 Framework

The overall framework of the novel-slot pipeline consists of a classification model, training objectives, and the detection algorithms. The classifica-

tion model consists of an embedder, encoder, and neural classification layer. The training objectives define the type of classification task (binary or multiple) at training time and test time. Finally, the detection algorithms determine which tokens are novel slots and are only used at test time.

4.1 Model Architecture

The model consists of three components: the embedding layer, encoder, and neural classifier.

Embedding Layer: The first component is a BERT-large (Devlin et al., 2018) uncased embedder layer which generates word embeddings from the query. BERT is a bidirectional encoder-based transformer that uses masked language modeling and next-sentence prediction for pretraining and language understanding (Devlin et al., 2018). The BERT-large embedder layer has three components: a token embedder, segment embedder, and positional embedder.

More specifically, let $\vec{s} = \{w_1, ..., w_n\}$ be a sentence. Prepend and append the sentence with $w_s =$ [CLS] and $w_e =$ [SEP] tokens respectively. Then, for all $i \in [n] \cup \{s, e\}$, w_i is first tokenized using a WordPiece (Wu et al., 2016) tokenizer T containing a vocabulary V of size 30,522 tokens (Devlin et al., 2018; Hasan, 2021). If $w_i \notin V$, then w_i is set to a unique out-of-vocabulary token [UNK]. Finally, w_i is indexed by T into $t_i = T(w_i)$ between 0 and 30,522.

Let E, S, and P be the token, segment, and positional embedders respectively. The token embedder created embeddings for the individual tokens while the positional embedder is essential for capturing the relative position of each token in the sequence. Although not relevant for our task, the segment embedder is needed to separate sentences. Thus, each t_i from tokenization is used to get three resulting embeddings, all of size 1024.

$$\mathbf{e_i} = E(t_i), \mathbf{s_i} = S(t_i), \mathbf{p_i} = P(t_i)$$

The three embeddings are added element-wise to produce the final embedding $\mathbf{o_i} \in \mathbb{R}^{1024}$

$$\mathbf{o_i} = \mathbf{e_i} + \mathbf{s_i} + \mathbf{p_i}$$

In short, the BERT embedding layer takes as input a sentence $\vec{s}$ and outputs an embedding $\mathbf{o_i}$ for each $w_i \in \vec{s}$ (that is, for each token). These embeddings are then used as input for the encoding layer.

Encoding Layer: The encoding layer consists of a bidirectional LSTM layer paired with a conditional random field, or simply BiLSTM-CRF (Huang et al., 2015) for short. Let B and F be the BiLSTM and CRF respectively. B has an input size of 1024 and a hidden size of 128. B takes as input the embeddings $\mathbf{o_i}$ in a sequential fashion in both directions. In the end, each word w_i in $\vec{s}$ will have an encoding l_i. The encodings are then passed as input into the conditional random field, which employs a softmax function and the Viterbi algorithm for inference.

Importantly, the use of a BiLSTM for encoding allows the model to learn long-range contextual dependencies from both directions. Additionally, the use of a CRF allows for the flexible design of features that are useful or significant for inference. Indeed, Li (2020) explains that having a CRF on top of a BiLSTM layer enables adding constraints to the outputs, which drastically decreases the number of incorrectly predicted labels. In the context of novel slot detection, one such constraint could be to incorporate a feature that forces the first token to be a "B-slotType" or "O" tag (Li, 2020).

The embedder and encoder are trained on in-domain training data to predict only in-domain slot types. At test time, detection algorithms are used to determine novel slots.

Neural Classifier Layer: The neural classifier layer is a neural network that is placed on top of the encoder used to predict in-domain slot types at test time. In particular, let l_i be an encoding generated by the encoding layer. Then, the neural classifier will assign the most likely tag for l_i. Eventually, the sentence $w_1, ..., w_n$ will be assigned a most likely sequence of tags $y_1, ..., y_n$, respectively.

4.2 Training Objectives

The two training objectives used in this paper are multiple and binary classifier. The multiple classifier classifies each token as a 'B-slotType', 'I-slotType', or 'O' tag. In contrast, the binary classifier classifies each token as an 'O' or 'non-O' tag. Only the multiple classifier is used on IND tagging whereas the binary and/or multiple classifier is used to predict novel slots. This is because the

binary classifier is proposed to improve identification of null 'O' tags specifically separating them from novel (and in-domain) slots (Wu et al., 2021).

4.3 Detection Algorithms

Two detection algorithms were employed at test-time to determine whether a token should be considered part of a novel slot: Maximum Softmax Probability (MSP) and Gaussian Discriminant Analysis (GDA).

Maximum Softmax Probability: The MSP method labels a token as a novel slot (or NS) if the maximum softmax probability falls below a given threshold. Formally, let w_i be a token and $\vec{z}$ be a vector of logits for w_i distributed over all slot types. Also, let F be the MSP algorithm. If we choose a threshold γ, then:

$$\max(\sigma(\vec{z})) < \gamma \implies F(x_i) = NS$$

Intuitively, MSP denotes a token to a novel slot if there does not exist an obvious best candidate among the set of in-domain slots to which the token can be assigned.

Gaussian Detection Algorithm: In the GDA algorithm, each slot type has a representation cluster in a Euclidean space. Then, given a new token, GDA assigns it to a novel slot if the minimum distance between the token and any representation cluster is greater than some threshold.

Formally, let w_i be a token and C be any representation cluster. Also, let F be the GDA algorithm and D be a function that computes the minimum distance between a token and a cluster. If we choose a threshold ϵ, then:

$$\min_C(D(x_i, C)) > \epsilon \implies F(x_i) = NS$$

Intuitively, the closer a token is to a cluster C, the more likely it is for the token to be a member of C. However, if the shortest distance to any cluster is large, this suggests that the token may not belong to a known cluster; hence, GDA assigns it as a novel slot.

In Wu et al. (2021), the authors also used a difference distance strategy for thresholding. Namely, GDA would label a token as a novel slot if the difference between the maximum and minimum distance to a cluster was less than the threshold ϵ.

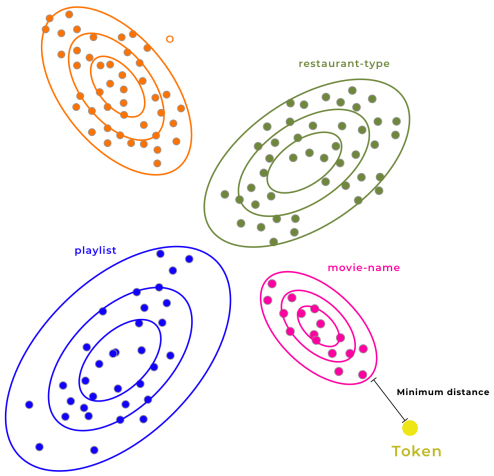

Figure 1: Representation clusters for four different slot types are shown along with a new token in a Euclidean space. In the minimum distance strategy, the GDA algorithm labels the token as a novel slot if the minimum distance to a cluster is greater than some threshold.

5 Evaluation

Multiple evaluation metrics were considered on both the token and span levels. Span F1 is the metric used most in the original slot filling task, but in the new NSD task, its rigidity regarding the beginning and end of slots is less suitable since the identification of novel slots is more important practically than the assignment of the beginning and end of the slot. Using Token F1 addresses this issue by considering the tags for each token individually (Wu et al., 2021). Due to the novelty of the NSD task, a new metric restriction-oriented span evaluation (ROSE) has been proposed to better emulate the Span F1 for NSD tasks without imposing strict ends on the identified novel slot. The ROSE metric considers a span correct when the number of correct token predictions within the span exceeds a predetermined proportion p. Thus, partially identified novel slots and novel slots predictions that exceed the correct length are still considered correct, avoiding the issue of imprecise but useful novel slot predictions (Wu et al., 2021).

6 Implementation

This section provides technical details on our baseline and modifications. All training was performed on one NVIDIA GeForce GTX 1080 Ti GPU with a batch size of 64. The tools used were the AllenNLP

library and PyTorch.

6.1 Baseline Replication

For baseline replication, we use a BERT-large model for the embedder and a BiLSTM-CRF for the encoder. The model is trained on all three datasets. More details are provided in Section 7.

6.2 Modification 1: Parameter Adjustments

From the baseline, we observed that the validation loss started increasing significantly at the end of training. This motivated us to test different values for dropout and patience to observe whether final validation loss could be decreased. The original model uses a dropout of 0.5 and a patience value of 10 epochs.

6.3 Modification 2: Token Embedder

In our second modification, we change the model used for token embeddings from BERT to ELMo (Peters et al., 2018) and the OpenAI transformer (Radford et al., 2018).

First, we evaluate model performance when using the ELMo model to generate embeddings. Specifically, we change the token indexer to an ELMo-specific token indexer, which converts a token to an array of indices, and we replace the BERT layer with an ELMo layer. At a high-level, ELMo uses stacked language models, namely LSTMs, to generate contextualized word embeddings (Peters et al., 2018). The stacked language models run in both the forward and backward directions, allowing the model to make use of context from both the "past" and "future." ELMo takes a weighted average of the input states of layers to capture both syntactic and semantic meaning. Notably, ELMo concatenates forward and backward vectors whereas BERT uses masked-language modeling and next-sentence prediction to generate word embeddings with whole context. Since BERT embeddings integrate context from both directions, we hypothesize that the BERT model will outperform ELMo.

Next, we consider using the OpenAI transformer (which we denote simply as OpenAI) to generate embeddings. Here, we use a OpenAI-specific token indexer and replace the BERT layer with an OpenAI layer. The OpenAI model consists of 12 stacked decoders and is pre-trained using thousands of books in an unsupervised fashion (Radford et al., 2018). Importantly, OpenAI is an autoregressive model that trains by predicting next tokens. Like with ELMo, the purpose of this modification is to confirm whether using BERT for embeddings is in fact the preferrable alternative over OpenAI. Because OpenAI is unidirectional, we conjecture that BERT will outperform OpenAI.

6.4 Modification 3: Seq2Seq Encoder

In our last modification, we change the type of Seq2Seq encoder from an LSTM to (1) a GRU, (2) a Multi-Head Self-Attention (Multi-Head) encoder, and (3) a Stacked Self-Attention (Stacked) encoder.

GRUs have fewer parameters than LSTMs and take less time to train and test. In cases where dataset size is small, GRUs tend to perform better. Since our dataset size is fairly small with 13,084 train utterances, we change the encoder type to GRUs to observe the effect on performance.

With the introduction of transformers (Vaswani et al., 2017), we know that attention can replace and outperform RNN-based architectures. This motivated us to test Multi-Head and Stacked encoders, which use an attention mechanism.

To change the encoder, we modify the encoder configuration files as well as the RNN initialization for the NSD slot tagging model.

7 Baseline Replication

We reproduce two models of the baseline results from the Wu et al. (2021) paper: GDA + Multiple + Minimum (GMM) and GDA + Multiple + Difference (GMD) where each model name is in the form "{Detection Method} + {Classifier Type} + {Distance Strategy}." The code for GMM was provided by the authors of the original paper, and we separately implemented the difference distance strategy for GMD.

In addition to the Token F1 and ROSE evaluation metrics code provided by Wu et al., we implemented Span F1 to observe how Span F1 measures from the modifications compared to the published benchmarks in Wu et al. (2021). The code for Span F1 was based on the provided Token F1 code.

See Table 2 for baseline results from Wu et al. (2021) and the reproduced results for each model. See Figures 2 and 3 for visual representations of the results for the NSD task. We observe that our reproduced results differ considerably from the paper results for the NSD task but are reasonably similar for the IND task (displayed in Appendix B).

Models			5%			15%			30%		
			IND	NSD		IND	NSD		IND	NSD	
detection method	objective	distance strategy	Span F1	Span F1	Token F1	Span F1	Span F1	Token F1	Span F1	Span F1	Token F1
GDA	multiple	difference	**92.50** 93.14	**25.62** 29.73	**45.18** 45.99	**86.42** 90.07	**15.19** 31.96	**35.98** 53.02	**88.11** 85.56	**15.16** 36.16	**36.88** 54.55
		minimum	**92.50** 93.14	**25.62** 29.73	**45.18** 45.99	**86.42** 90.07	**15.19** 31.96	**35.98** 53.02	**88.11** 85.56	**15.16** 36.16	**36.88** 54.55

Table 2: For each model and Snips dataset, the reproduced results are displayed in blue and the corresponding results from Wu et al. (2021) are displayed in green.

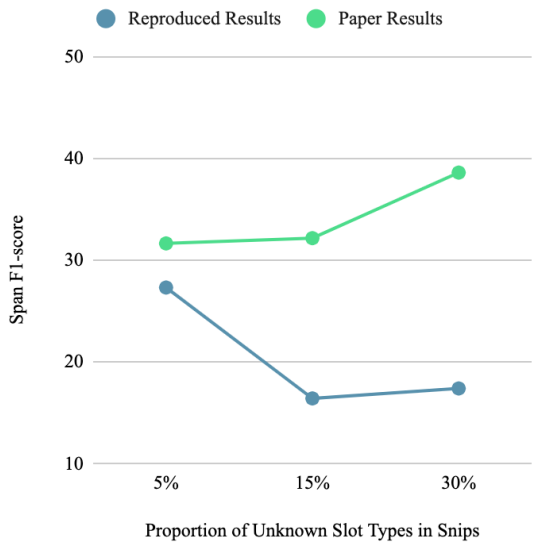

Figure 2: Results for the GMM model on the NSD task with the Snips dataset.

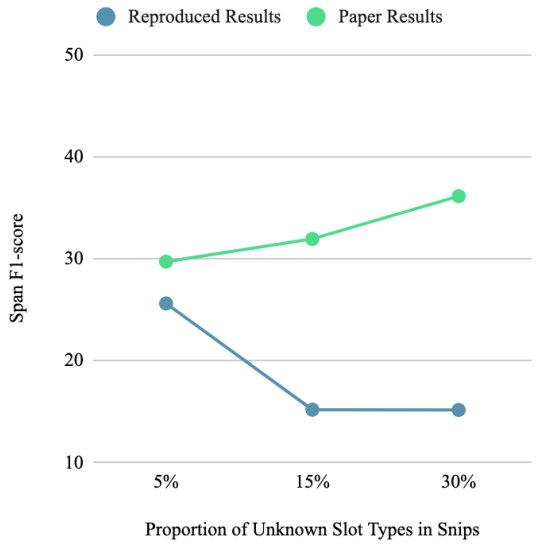

Figure 3: Results for the GMD model on the NSD task with the Snips dataset.

8 Modification Results

All modifications are performed on the GMM model provided by Wu et al. (2021). After training, we evaluate the models on the test set using the metrics described in Section 5. Quantitative results are presented in Tables 4 and 5 in the Appendix. We focus more on performance in the NSD task, but we still include results on the IND task to observe how our changes affect performance for IND.

8.1 Modification 1: Parameter Adjustments

See Table 6 for the results using different dropout and patience values. We increase and decrease the dropout from its original value of 0.5. Table 6 displays the results for dropout values of 0.4, 0.6, and 0.8. All changes produce a lower performance. We also change the patience from its original value of 10 and observe no improvement in performance.

8.2 Modification 2: Token Embedder

As Tables 4 and 5 depict, the original model outperforms both the models with the ELMo and OpenAI embedders in terms of IND Span F1, NSD Span F1, and NSD Token F1 across all datasets. For ELMo, we observe that as the novel slot proportion increases (i.e., transition from Snips5% to Snips30%), the IND Span F1 and NSD Token F1 discrepancies between the original and ELMo model decrease. In particular, the percent decrease in IND Span F1 from the original to the modified model was 8.07%, 6.06%, and 5.30% for 5%, 15%, and 30% datasets respectively. For NSD Token F1, these percent decreases were 55.37%, 35.77%, and 28.42% for 5%, 15%, and 30% datasets respectively. We observe that BERT outperforms ELMo and OpenAI for the ROSE and Span F1 metrics. See Figure 4 for the results on Snips15% and Appendix C for the results on the other datasets.

8.3 Modification 3: Seq2Seq Encoder

In modification 3, we change the LSTM encoder to GRU, Multi-Head, and Stacked encoders. See

Figure 5 for the results on Snips15% and Appendix D for the results on the other datasets. We observe that using a GRU generally does not improve performance, though it produces slightly higher scores for the Snips15% dataset. Multi-Head performs consistently worse than the LSTM model, with the exception of the ROSE values, which are relatively similar for each dataset. For Stacked in the Snips15% dataset, as the proportion of novel slots increases in the Snips dataset, the performance gets increasingly better than that of LSTM.

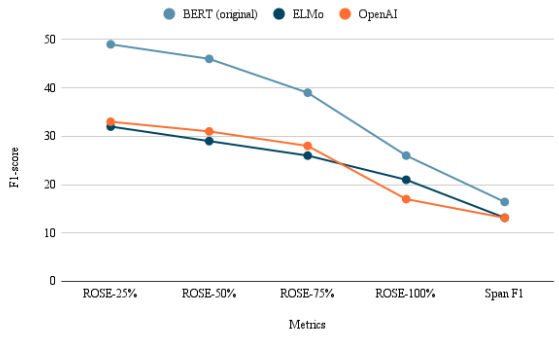

Figure 4: Results for modifying the token embedder on the Snips15% dataset.

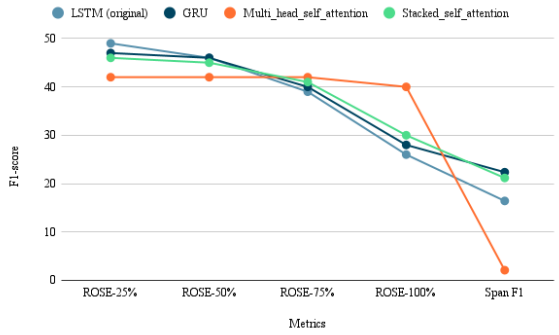

Figure 5: Results for modifying the Seq2Seq encoder on the Snips15% dataset.

9 Discussion

9.1 Reproducing Baselines

The differences between our reproduced results and the Wu et al. (2021) results for the NSD task may be due to differences in the method of calculating the threshold values used for the distance strategy. We were unable to exactly reproduce the Wu et al. results because the specific threshold values that the authors used were not disclosed. In that paper's example for training, the authors' provided threshold value is 8, but this value did not produce the exact same results as in the paper. Since the Wu et al. paper details that threshold values were adaptively calculated using the best F1 scores on the validation set but did not provide a calculation method, we trained on each dataset using multiple threshold values to find the threshold value that resulted in the closest values compared to the original authors' results. We started with the threshold value of 8 (since it was used in the example) and tested values close to 8, increasing or decreasing the value based on whether the results were closer or further from the published results in Wu et al. (2021).

In Table 3, we display the final threshold values we found to produce results that were closest to the published results for each baseline model and for each of the Snips datasets.

	GDA+Multiple+Minimum	GDA+Multiple+Difference
SNIPS5%	8.1	5.1
SNIPS15%	7.8	6.0
SNIPS30%	8.2	6.5

Table 3: Threshold values that yielded baseline results closest to the results published in Wu et al. (2021).

9.2 Modification 1: Parameter Adjustments

Since validation loss increased near the end of training, we tested lower dropout and patience values for the model. None of the experiments yielded higher performance. Thus, we confirm that the original parameter values were correctly chosen in the Wu et al. (2021) paper so that further adjustments may not be necessary. Note that it also makes sense that some increase in validation loss at the end of training does not necessarily indicate poor prediction accuracy overall because it may indicate having just passed an optimal point in testing accuracy before reaching catastrophic overfitting.

9.3 Modification 2: Token Embedder

BERT outperformed ELMo across all the metrics as we predicted. We propose several potential contributing factors. First, unlike BERT, ELMo does not consider contexts in both directions simultaneously but rather simply concatenates states from the forward and backward passes. In addition, unlike BERT, ELMo is not transformer-based, so it does not have the advantages of a built-in attention mechanism. Collectively, this suggests that ELMo is a less powerful model than BERT for Novel Slot Detection, a task that uses bidirectional context.

BERT also outperformed OpenAI across all metrics. We suggest that this is because OpenAI is a unidirectional, autoregressive model. Notwithstanding, this makes OpenAI effective for generative tasks but less effective on Novel Slot Detection in which bidirectionality has proven useful. To understand how bidirectionality is useful, consider the sentence "Play Hello by Lionel Richie" and suppose song is an in-domain slot type. In confining the direction of context, OpenAI deprives itself of the ability to use the "Lionel Richie" for context to properly classify "Hello" as a song.

We also observed that for both ELMo and OpenAI, the novel-slot proportion and discrepancies with the original model were inversely proportional. We believe that for NSD Token F1, this is because increasing the novel-slot proportion makes it more likely to correctly predict novel slots. For IND Span F1, we conjecture that perhaps a higher novel slot makeup makes it more difficult to identify the boundaries of in-domain spans since the sample is more likely to contain a nearby novel slot.

9.4 Modification 3: Seq2Seq Encoder

We observe that the GRU model performs better than the original model on the Snips15% dataset on some metrics in Figure 5. However, on the other datasets, the GRU model performs worse. This may be because our inputs are short since GRUs perform better on data with long inputs but small datasets (Yang et al., 2020).

The Multi-Head model applies the attention mechanism to the encoder allowing the model to focus on multiple attributes, or "heads", when identifying relationships between tokens. Since the samples in our dataset are short, it makes sense that the model does not benefit from multiple attention heads. Thus, the ROSE metric values are all relatively the same, in contrast with the other encoders as seen in Figures 5, 10, and 11. With short sequence length from our data input, multiple heads may actually distract the model from locating meaningful attributes when detecting a novel slot, which may be why the Span F1 is significantly lower compared to the other encoders.

The Stacked model applies self-attention in multiple layers, which is a more complex encoder. Stacked includes a 2-layer feedforward network, multi-head self-attention, and layer normalisation. These components are stacked in a specified number of layers, which we set to the default of 2; this depth may be why Stacked is more effective than the other encoders at detecting high proportions of novel slots. Thus, the Stacked model shows an increase in performance as the proportion of novel slots in the dataset increases from 5% to 30%.

10 Conclusion and Future Work

This paper investigates the choice of model framework and hyperparameters in the proposed novel slot detection task by Wu et al. (2021). We determine that BERT bidirectional embeddings and an LSTM encoder are suitable for a base framework to establish a benchmark for the NSD task, as shown by the consideration of alternative embedders and encoders. Additionally, since the results for the Stacked Self-Attention encoder show an improvement in performance as the proportion of novel slots increases in the dataset, we observe that Stacked Self-Attention based models have potential to improve the basic NSD model for data with a high proportion of novel slots.

We suggest that future work employs more powerful bidirectional embedders such as RoBERTa (Liu et al., 2019) or XLNet (Yang et al., 2019) since this work establishes the superiority of results from bidirectional embeddings. Specifically, this paper establishes that autoencoder models (namely BERT) outperform purely autoregressive models (ELMo and OpenAI) as can be expected since autoencoder models are able to use bidirectional contexts. Thus, analyzing how other bidirectional embedders perform could improve model performance. In particular, we hypothesize that RoBERTa and XLNet embedders will outperform the benchmark since RoBERTa was trained using significantly more data and time than BERT (Liu et al., 2019) and XLNet combines the advantages of both autoregressive and autoencoder language models (Yang et al., 2019).

Additionally, similar evaluations of the proposed framework on additional NSD datasets such as ATIS-NSD should be explored (Wu et al., 2021; Hemphill et al., 1990). Finally, changes to the proposed evaluation metrics should be considered to establish the ROSE metric as the standard for NSD.

Code Availability

All code from this paper can be accessed here. This repository contains files from the original repository (Wu et al., 2021) and the files that we implemented.

Acknowledgments

We would like to thank our project adviser, Carlos Jimenez, for his guidance, as well as course staff and fellow students for feedback at the poster session.

References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibault Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Soyeon Caren Han, Siqu Long, Huichun Li, Henry Weld, and Josiah Poon. 2022. Bi-directional joint neural networks for intent classification and slot filling.

Mohd Shukri Hasan. 2021. Why bert has 3 embedding layers and their implementation details.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, abs/1610.02136.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Maolin Li. 2020. Crf layer on the top of bilstm - 1.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Henry Weld, Xiaoqi Huang, Siqi Long, Josiah Poon, and Soyeon Caren Han. 2021. A survey of joint intent detection and slot-filling models in natural language understanding. *CoRR*, abs/2101.08091.

Yanan Wu, Zhiyuan Zeng, Keqing He, Hong Xu, Yuanmeng Yan, Huixing Jiang, and Weiran Xu. 2021. Novel slot detection: A benchmark for discovering unknown slot types in the task-oriented dialogue system. *CoRR*, abs/2105.14313.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Hong Xu, Keqing He, Yuanmeng Yan, Sihong Liu, Zijun Liu, and Weiran Xu. 2020. A deep generative distance-based classifier for out-of-domain detection with mahalanobis space. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1452–1460, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Shudong Yang, Xueying Yu, and Ying Zhou. 2020. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, pages 98–101.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

A Quantitative Results

Modifications		5%			15%			30%		
		IND	NSD		IND	NSD		IND	NSD	
		Span F1	Span F1	Token F1	Span F1	Span F1	Token F1	Span F1	Span F1	Token F1
Original		**92.74**	**27.33**	**57.87**	85.19	16.42	41.43	87.62	17.40	43.53
Token Embedder	ELMo	85.26	8.69	25.83	80.03	13.15	26.61	82.98	8.57	31.16
Seq2Seq Encoder	GRU	91.29	13.03	39.52	**86.09**	**22.36**	37.54	**88.26**	13.35	32.26
	Multi-Head	18.64	1.17	10.71	14.36	2.10	19.33	23.90	4.27	22.29
	Stacked	87.41	9.99	38.42	81.03	21.17	**42.49**	86.53	**22.03**	**49.42**

Table 4: This table displays the results for modifying the token embedder and the Seq2Seq encoder. The original model uses BERT for the token embedder and LSTM for the encoder. Best numbers are bolded across all metrics.

Modifications		15%		
		IND	NSD	
		Span F1	Span F1	Token F1
Original		**85.19**	**16.42**	**41.43**
Token Embedder	OpenAI	83.49	13.14	29.26

Table 5: This table displays the results for replacing the BERT embedder with OpenAI transformer for the Snips15% dataset. Best numbers are bolded across all metrics.

Modifications		30%		
		IND	NSD	
		Span F1	Span F1	Token F1
Original		**87.62**	**17.40**	**43.53**
Dropout	0.4	77.15	1.07	24.64
	0.6	80.76	1.54	27.89
	0.8	67.20	0.54	23.39
Patience	1	79.85	2.49	28.81
	2	80.87	1.94	26.87
	5,8,9,11*	81.02	1.50	26.99

Table 6: This table displays the results for different dropout and patience values. The original model uses a dropout of 0.5 and a patience of 10. Best numbers are bolded across all metrics.

*Patience values of 5, 8, 9, and 11 produced the same results.

B Baseline Replication Results for IND

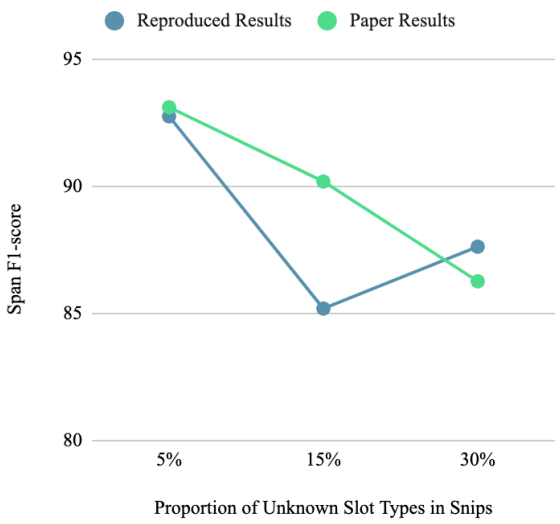

Figure 6: Results for the GMM model on the IND task with the Snips dataset.

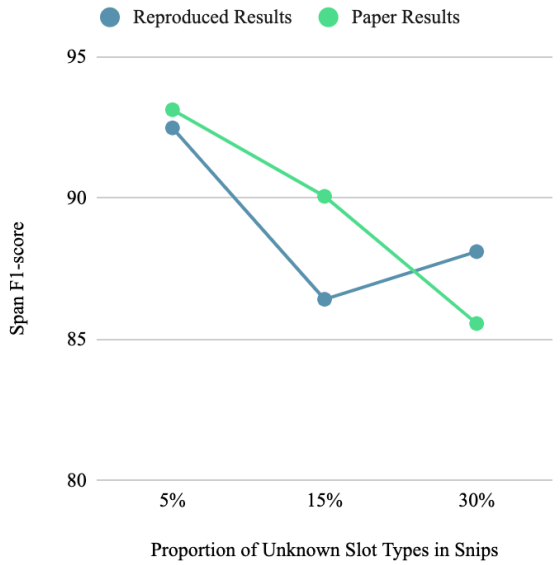

Figure 7: Results for the GMD model on the IND task with the Snips dataset.

C Results for token embedders on Snips5% and Snips30%

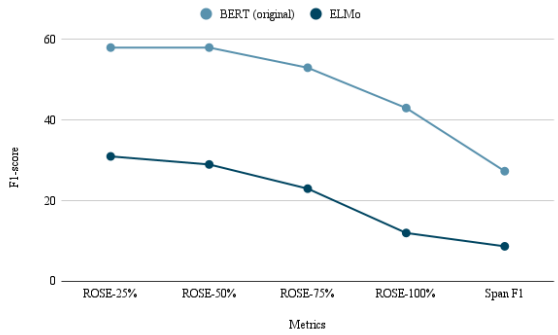

Figure 8: Results for modifying the token embedder on the Snips5% dataset.

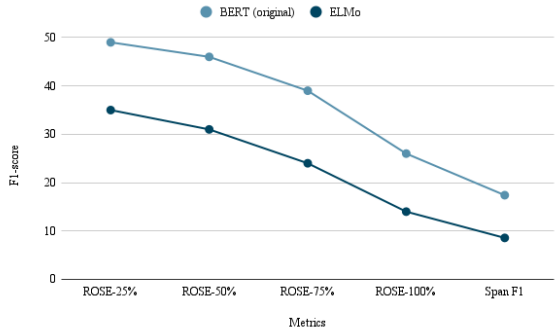

Figure 9: Results for modifying the token embedder on the Snips30% dataset.

D Results for encoders on Snips5% and Snips30%

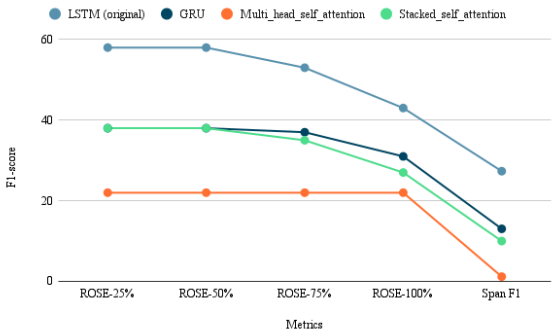

Figure 10: Results for modifying the Seq2Seq encoder on the Snips5% dataset.

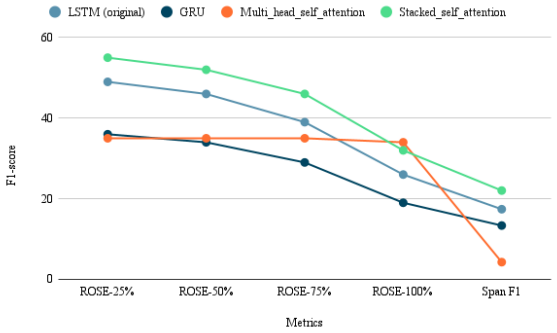

Figure 11: Results for modifying the Seq2Seq encoder on the Snips30% dataset.