IntentBot: Building Machine Learning Systems For Automated Intent Detection

Robert Yang (bobyang9@cs.stanford.edu), Tai Vu (taivu@stanford.edu)

Problem and Motivation

Motivation: Intent detection is a highly needed service spanning numerous industries. AI-powered task-oriented dialog systems can automate customer service by classifying
requests from emails, social media, or texts into categories such as "Bug Report", "Question", and "Purchase", thus boosting business productivity.

Problem: This project aims to classify intents using machine learning and deep learning. The input to our algorithm is a sentence indicating a user command. Our models then output a predicted intent label.

Solution: Naive Bayes, Softmax Regression, SVM, Bi-LSTM, and BERT.

Data and Input Pipeline

Dataset:
- SNIPS contains day-to-day commands like “get weather” and “play music”, with 13802 training samples and 699 test samples. 7 user intents.
- Kaggle ATIS): English air travel queries such as “airfare” and “ground service”. 4977 training examples and 893 test examples, which belong to 8 intents.

Input	Label
please play me a top nineties theme song	PlayMusic
where can i purchase the video game the blue generation	SearchCreativeWork
what will the weather be in uzbekistan at 4 am	GetWeather

Table 1. Examples in the SNIPS dataset.

Input pipeline:
- Converted raw data files into input-output pairs.
- Removed punctuation, turned sentences into lowercase
- Tokenized each input.
- Split data into 95% (training) and 5% (validation)
    - SNIPS: 13112 in training set and 690 in validation set
    - ATIS: 4728 in training set and 249 in validation set

Features:
- Bag of Words: Feature vector representing occurrences of words in a dictionary.
- Pre-trained neural embeddings: GloVe and BERT.

Models

Naive Bayes (Bernoulli event model)

$$p(x_j = 1 \mid y = k) = \frac{\sum_{i=1}^{n} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = k\}}{\sum_{i=1}^{n} 1\{y^{(i)} = k\}}$$

$$p(y = k) = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = k\}}{n}$$

Models, cont.

Naive Bayes (Multinomial event model)

$$p(x_j = l \mid y = k) = \frac{1 + \sum_{i=1}^{n}\sum_{j=1}^{d_i} 1\{x_j^{(i)} = l \wedge y^{(i)} = k\}}{|V| + \sum_{i=1}^{n} 1\{y^{(i)} = k\} d_i}$$

$$p(y = k) = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = k\}}{n}$$

Softmax Regression
- A generalization of the Logistic Regression algorithm for multi-class datasets
- Inference: $\frac{exp(W_i^T x + b_i)}{\sum_{j=1}^{k} exp(W_j^T x + b_j)}$
- Training: $W = W - \alpha(\hat{y} - y)X$
  $b = b - \alpha \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})$

SVM (Support Vector Machine)
- Fits the data through maximizing the margin of separation between different classes
- Inference: $h_{w,b}(x) = g(w^T x + b)$
- Optimization objective: $min_{\gamma,w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$
  so that $y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$ for $i = 1, ..., n$
  and that $\xi_i \geq 0$ for $i = 1, ..., n$
- Used multi-class variant of SVM where a model is created to distinguish every pair of two classes

LSTM (Long Short-Term Memory)
- A model for processing sequences of data;
- Based on the recurrent neural network (RNN): The hidden state at the previous timestep, along with the new input at the current timestep, can influence the current hidden state;
- LSTM has a cell memory with three gates (forget gate, input gate, and output gate), to preserve information over long sequences;
- Used Bidirectional LSTM (Bi-LSTM), which involved two LSTM models trained in opposite directions so that more context is provided.

BERT (Bidirectional Encoder Representations From Transformers)
- Encoder use Transformer architecture, outputting a 768 dimensional embedding for each input sentence.
- A fully connected layer and a softmax layer followed the encoder.
- Pre-trained on a large corpus and fine-tuned on ATIS and SNIPS.

Experiments and Results

Algorithm	Features	Val Acc	Val. F1
Naive Bayes (Bernoulli)	Bag of Words	0.9783	0.9782
Naive Bayes (Multinomial)	Bag of Words	0.9739	0.9737
Softmax Regression	Bag of Words	**0.9846**	**0.9869**
Softmax Regression	GloVe	0.9725	0.9721
Softmax Regression	BERT	0.9667	0.9663
Support Vector Machines	Bag of Words	0.9290	0.9291
Support Vector Machines	GloVe	0.9797	0.9797
Support Vector Machines	BERT	**0.9812**	**0.9812**
LSTM	One-hot	**0.9812**	**0.9811**
LSTM	GloVe	0.9609	0.9610
LSTM	BERT	0.9696	0.9696
BERT	N/A	**0.9913**	**0.9913**
Naive Bayes (Bernoulli)	Bag of Words	0.9256	0.9247
Naive Bayes (Multinomial)	Bag of Words	0.9256	0.9236
Softmax Regression	Bag of Words	0.9587	0.9552
Softmax Regression	GloVe	0.9132	0.8940
Softmax Regression	BERT	**0.9669**	**0.9686**
Support Vector Machines	Bag of Words	0.9215	0.9079
Support Vector Machines	GloVe	0.9463	0.9397
Support Vector Machines	BERT	**0.9628**	**0.9568**
LSTM	One-hot	**0.9711**	**0.9692**
LSTM	GloVe	0.9132	0.8890
LSTM	BERT	0.8967	0.8960
BERT	N/A	**0.9959**	**0.9962**

Table 2. Performance comparison of different models (upper: SNIPS, lower: ATIS)

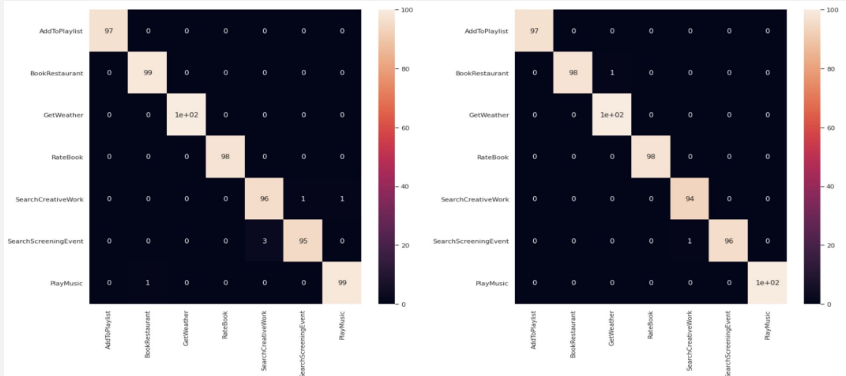

Figure 1. Confusion matrix for BERT (left) and LSTM (right).

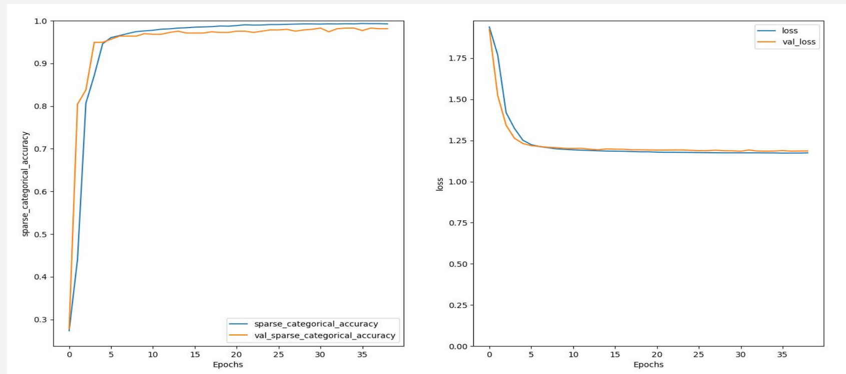

Figure 2. Accuracies (left) and Losses (right) on training set (blue) and validation set (orange) when Training Bi-LSTM.

Discussion

- BERT performed the best, with 99+% accuracy and 0.99+ F1 scores. Its Transformer architecture had the attention mechanism, which allowed it to identify important parts of the text (even those far away), which resulted in informed classifications.
- Softmax Regression with Bag of Words feature extraction, a much simpler model, came pretty close in performance. It performed better than Bi-LSTM, a complex deep learning model. Other algorithms like Naive Bayes and SVMs also had great performance despite their simplicity..
- Fine-tuning the BERT model leads to a better outcome than combining a fixed pre-trained BERT or GloVe encoder with a classifier. It is because the distribution of words in the pre-trained English corpora and in our intent datasets are different (the ATIS and SNIPS examples only belong to limited domains). Therefore, the fine-tuning process updates the weights in BERT encoder to adapt better to our training sets.

Conclusion and Future Work

Summary:
BERT model performed the best, with 99+% accuracy and 0.99+ F1 scores on test data.

Next Steps:
- Fine-tuning hyperparameters.
- Evaluate our algorithms on CLINC150.
- Build models for joint intent detection and slot filling.
- Deploy the models in chatbot.

References

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.