# ATIS slot filling - RNN

Chrysomallis Iason and Papadopoulos Argiris.
*Technical University of Crete.*
(Dated: June 24, 2019)

**ABSTRACT**

The problem we aim to tackle in the current work is a natural language problem, which aims to extract meaning of speech utterances. Specifically we want to present an automated system, in which the model receives sentences from callers (who require some sort of service, i.e. ticket purchase from an airline company) and distinguishes semantic words from irrelevant ones. This problem can be solved by training a Deep Neural Network model to extract this useful information.

## I. INTRODUCTION

With the wide variety of Deep Learning models and their use for quality of human life, it is fair to say that problems such as service complexity can be reduced plenty. Specifically we offer a solution in which callers, who want to purchase an airline ticket via phone, ask for information, and a model receives sentences by speech recognition and evaluates words' relevance. To create this type of sequential model, ATIS slot filling data provides a reliable set in which the model is trained to offer this type of solution. Multiple types of models are presented in the current work, RNN, LSTM, Convolution1D, such as their respective results.

## II. ATIS DATA SET

The Airline Travel Information System (ATIS) data set, that was collected by DARPA in the early 90s, is split into 5 folds, and each fold contains train and test data, for training and evaluating respectively, as well as labels to each and every sentence for weight propagation and score computation. Also dictionaries are contained, so the data/labels can be mapped to words and vice-versa. For example the sentence *"Show flights from Boston to New York today."* is reduced to identifying arguments like *Destination* and *Departure* and this task is called slot-filling.

| **Words** | Show | flights | from | Boston | to | New | York | today |
|---|---|---|---|---|---|---|---|---|
| **Labels** | O | O | O | B-dept | O | B-arr | I-arr | B-date |

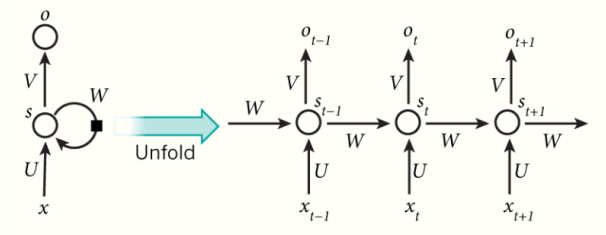
There are 3983 training sentences and 893 testing sentences per fold with a vocabulary size of 572 words and 127 labels (including the NULL symbol 'O').

## III. WORD EMBEDDINGS - KERAS RNN MODEL

**3.1** Word Embeddings map words to a high-dimensional vector and they can learn semantic and syntactic information of the words. As an illustration the nearest neighboors of some words are presented.

| | | | | |
|---|---|---|---|---|
| sunday | boston | august | time | car |
| wednesday | nashville | september | schedule | rental |
| saturday | toronto | july | times | limousine |
| friday | chicago | june | schedules | rentals |
| monday | phoenix | december | dinnertime | cars |
| saturdays | columbus | january | departure | ap |

**3.2** Recurrent Neural Networks prove a reliable solution for this type of sequential problem because if learnt correctly, they can predict words based on the previous ones. RNNs have internal memory which store the summary of the sequence so far. This slot-filling task is a complicated word tagging problem that requires such a model which stores and remembers the word sequence.




1. $x_1, x_2, ..., x_{t-1}, x_t, x_{t+1}$ is the input to the RNN, in this case the **Word Embeddings**.
2. $s_t$ is the hidden state of the RNN at step t. This is computed based on the state at the step $t-1$ as $s_t = f(Ux_t + Ws_{t-1})$, where $f$ is a **softmax** classification function.
3. $o_t$ is the output of the model at step t and $o_t = f(Vs_t)$
4. $U, V, W$ are the learnable parameters of the model.

## IV. TRAINING - EVALUATION - RESULTS

For the training purpose, 3 models were used in total for comparisons based on the results they provided. First a simple RNN layer was used, as described above, then a more suitable model for a problem like this, a Long Short-Term Memory (LSTM) network which has modules such as input, output and forget gate that regulate the flow of information inside it's cell, and lastly an improved model

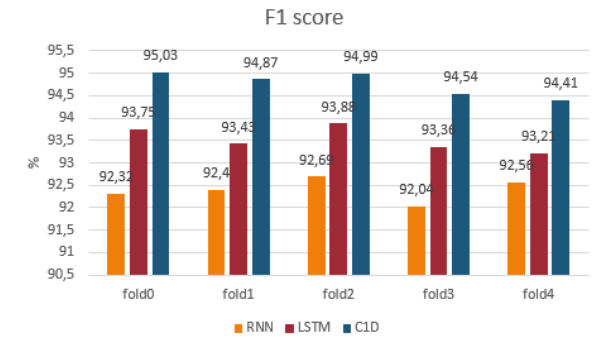
with a convolutional layer before RNN and after Word Embeddings. With the first and second model, one drawback was presented. There was no lookahead, the output depended only on the current and previous word, but not the following one. So this is fixed by taking advantage of convolutional layer properties.

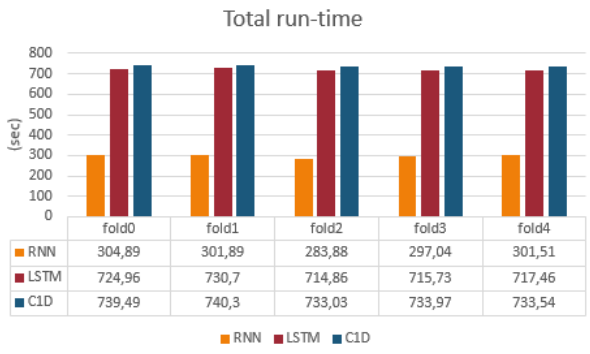
Because the number of samples in our data set is very small, the use of folds comes in handy for 5-fold cross validation. Also each batch is shuffled randomly before training to avoid repetition. 30 epochs deemed enough to extract representative results, as will be shown later, some models are time consuming. **Softmax** is the activation function for every layer for each model except the convolution layer is activated by **relu** function.

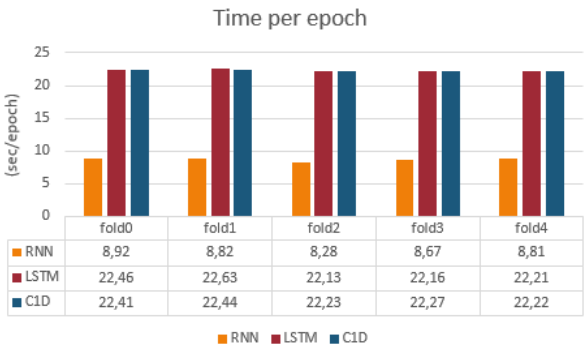
Accuracy is not the best performance criterion for this type of problem because of the small size of the data set, as good predictions can come from bad samples, which can ruin the learning process and give the false idea of a well constructed model. We use f1 score as a metric for performance which is computed as follows:

$F_1 = 2 \cdot \frac{p \cdot r}{p + r}$, $p$ as precision (when the model predicts positive, how often is it correct) and $r$ as recall (which helps when the cost of false negatives is high).

The following graphs present performance based on f1 score, total run-time and time per epoch while training on CPU version of python with specs: i5 6600 3.3GHz.




As seen above, the third model performs the best based on f1 score, followed by the model with LSTM layer and last comes the one with the simple RNN layer. It yields around 95% f1 which is reliable enough. Both the LSTM and improved model consume more than double the time to run for 30 epochs, around 12 minutes total and 22 sec/epoch, compared to the simple RNN layer model, which completes its process at around the 5 minute mark and 9 sec/epoch. With more powerful resources and further improvements parameter-wise (epochs, activation function selection etc.) a near perfect model can be created.

## V. CONCLUSION

It is fair to say that a natural language problem with the ATIS data set can be tackled using this very simple and fast implementation using RNN models and layers that store and process sequential data. With the rise of Dense Neural Networks plenty of problems like reducing customer service complexity by identifying semantic information automatically can be solved and raise the quality of human life.

## VI. CITATIONS


1. Grégoire Mesnil, Xiaodong He, Li Deng and Yoshua Bengio. Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding. Interspeech, 2013.

2. Gokhan Tur, Dilek Hakkani-Tur and Larry Heck. What is left to be understood in ATIS?

3. Christian Raymond and Giuseppe Riccardi. Generative and discriminative algorithms for spoken language understanding. Interspeech, 2007.

4. Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian, Bergeron, Arnaud, Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2012.

5. Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy), June 2010.