ATIS Challenge Report

Rodrigo de Oliveira

Introduction

This report describes the training, testing and a user-centred evaluation of a model to predict *intents* and *slots* in a flight-booking dialogue. For example, given the query:

“what is the cheapest flight going to new york city on march 24th in the morning”

The model should be able to detect that the **intent** of the utterance is *to enquire about flight details*, and the the **slot** “destination” would be filled with “new york”.

Implementation

PyText was the tool of choice to train the model, as it provides an easy-to-use interface to train models: a configuration file. The file *atis_config.json* has all instructions that PyText requires to train the model, from the machine learning algorithm to use, to paths to files or sizes (batches, epochs, etc). This is not only convenient for engineers training models for applications, but also for researches to share models and reproduce experiments. PyText is not provided with this report but installation instructions are given in *README.md*.

In order to see the model ‘in action’, a simple web application was implemented (*app.py*) which should be the barebones of a flight-flight-booking dialogue system. Being just a simple app to demonstrate the model, it does no more than output the predicted *intentions* and *slots* in the utterances you supply. For example, if you supply:

“what is the cheapest flight going to new york city on march 24th in the morning”

the application should return:

{
"intent": "flight",
"slots": "{'cost_relative': 'cheapest', 'toloc.city_name': 'new york city', 'depart_date.-month_name': 'march 24th', 'depart_time.period_of_day': 'morning'}"
}

The above means that the model recognised the intention of the utterance as being flight-related, and that certain slots of a flight booking form could already be filled:

- The name of the destination city: “new york”
- The departure date: “march 24th”
- The period of the day on the departure date: “morning”
- The relative cost of the flight: “cheapest”

The application is served with Flask, which is a very light-weight and fast application to serve other applications via RESTful APIs. Flask, however, is not production-ready, because it is intended for low traffic. WSGI servers (e.g. Waitress) are required for production environments. However, for demonstrative purposes, such as the above application, Flask alone should suffice. Training

The model supplied in this package (*atis_model.c2*) was trained with the following configuration and high parameters:

- **Joint task**: 2 separate labels exist in the training data—*intents* and *slots*—but both were used at the same time during training, so the model attempts to label new utterances with both label sets: 1 intent label for the whole utterance (so a document classification task) and 1 slot label per token (so a word classification/tagging task), although the same slot label can be assigned to multiple tokens.

- **LSTM**: RNNs such as LSTMs have the advantage of overcoming the vanishing gradient problem, which is very useful for sequential input such as text, because it can handle long-distance relations (very common in language). 2 layers were used.
- **Bidirectional**: so that the strings are processed both from left to right, but also from right to left. Long distance relations in text exist not only in past steps of the sequence (e.g. anaphora), but they may also exist in not yet seen parts of the sequence (e.g. cataphora).
- **Attention model**: so that the model can chose what to *attend* to, depending on what is more important. Dimension 128 was used.
- **Pre-trained word embeddings**: generic word vectors trained with simple and shallow models, but they help the model handle words that are not present in the training data. GloVe was used.

Testing

Table 1 shows how the model performs when predicting labels of whole utterances, i.e. the *intent* of utterances.

Table 1: Intent Scores

Per label scores	Precision	Recall	F1	Support
flight	98.26	98.58	98.42	632
flight_no	88.89	100	94.12	8
abbreviation	100	100	100	33
ground_service	97.3	100	98.63	36
airfare	96	100	97.96	48

airline	100	100	100	38
airport	94.44	94.44	94.44	18
city	75	100	85.71	6
capacity	95.45	100	97.67	21
ground_fare	100	100	100	7
distance	100	90	94.74	10
quantity	42.86	100	60	3
meal	83.33	83.33	83.33	6
aircraft	100	88.89	94.12	9
flight_time	50	100	66.67	1
airfare+flight	0	0	0	1
flight+airfare	80	33.33	47.06	12
day_name	0	0	0	2
flight_no+airline	0	0	0	1
flight+airline	0	0	0	1
Overall macro scores	**70.08**	**74.43**	**70.64**	

Table 2 shows how the table performs when predicting labels of individual tokens, i.e. the *slot* the token is supposed to 'fill' in a flight booking form.

Table 2: Slots scores

Perlabelscores	**Precision**	**Recall**	**F1**	**Support**
toloc.city_name	97.26	99.44	98.34	715
fromloc.city_name	98.87	99.57	99.22	703
fare_basis_code	94.44	100	97.14	17
booking_class	0	0	0	1
restriction_code	100	100	100	4
city_name	81.4	62.5	70.71	56
airline_code	94.29	97.06	95.65	34
toloc.airport_code	100	100	100	4
depart_date.day_name	98.61	97.26	97.93	146
airline_name	100	98.97	99.48	97

state_name	0	0	0	6
days_code	0	0	0	0
mod	0	0	0	2
flight_number	100	85.71	92.31	7
meal_code	100	100	100	1
meal	82.35	87.5	84.85	16
meal_description	100	75	85.71	8
aircraft_code	96.67	96.67	96.67	30
airport_name	75	60	66.67	20
airport_code	80	44.44	57.14	9
fromloc.airport_code	55.56	100	71.43	5
toloc.state_name	0	0	0	3
transport_type	90	90	90	10
class_type	94.74	100	97.3	18
depart_time.period_of_day	96.77	96.77	96.77	31
depart_date.today_relative	71.43	83.33	76.92	6
fromloc.airport_name	50	81.82	62.07	11
day_name	100	50	66.67	2
flight_stop	100	100	100	15
round_trip	100	94.59	97.22	37
flight_mod	78.26	78.26	78.26	23
fromloc.state_code	100	100	100	1
arrive_date.date_relative	50	100	66.67	1
arrive_date.day_name	63.64	77.78	70	9
toloc.airport_name	33.33	33.33	33.33	3
arrive_date.month_name	100	83.33	90.91	6
stoploc.city_name	86.36	95	90.48	20
flight_days	100	100	100	4
stoploc.airport_name	0	0	0	0
stoploc.airport_code	0	0	0	1
cost_relative	100	100	100	35
depart_time.time_relative	100	88	93.62	25
depart_date.month_name	95.83	97.87	96.84	47
flight_time	50	100	66.67	1

arrive_time.time_relative	80	85.71	82.76	14
depart_time.end_time	75	100	85.71	3
depart_time.time	25	33.33	28.57	3
arrive_time.period_of_day	71.43	100	83.33	5
compartment	0	0	0	1
connect	100	100	100	4
arrive_time.start_time	100	100	100	8
arrive_time.end_time	100	100	100	8
depart_time.start_time	75	100	85.71	3
depart_time.period_mod	100	100	100	1
arrive_time.time	62.5	100	76.92	5
flight	0	0	0	1
depart_date.date_relative	87.5	100	93.33	7
depart_date.day_number	100	100	100	1
fare_amount	100	100	100	1
return_date.day_name	0	0	0	1
period_of_day	0	0	0	1
return_date.date_relative	50	33.33	40	3
Overallmacroscores	**70.02**	**72.69**	**70.38**	

Overall, the model achieves a score of 71% F1 on intent prediction and almost the same (70% F1) on slots prediction. The full annotated corpus is distributed with this report at *docs/test_out.csv.*

Evaluating

In the *Test* section, we saw how the model achieves considerably high performance scores (70%+) when evaluated against a pre-annotated corpus. However, given that such a model is intended to be deployed in production, as part of an application to be used by *naive* users, preferably paying customers, a **user-centred evaluation** is mandatory. Table 3 shows *pseudo* Leikert scores (range 0-2, for simplicity, instead of 1-5 or

1-7) as judged by a real user of the application after interacting with the application (and the model within) over 21 queries.

Table 3

	0 = not satisfied	1 = partially satisfied	2 = satisfied
1			x
2		x	
3			x
4			x
5	x		
6	x		
7			x
8		x	
9			x
10	x		
11	x		
12	x		
13	x		
14		x	
15	x		
16	x		
17		x	
18	x		
19			x
20			x
21			x
total	9 (43%)	4 (19%)	8 (38%)

The user was fully satisfied with the model in only 1/3 of interactions (38%), but completely dissatisfied in the majority of scenarios (43%). The actual interactions (and assigned Leikert scores) can be at *docs/user_session.txt*.

Conclusion

This report described the training, testing and a user-centred evaluation of a model to predict intentions utterances and slot filling parts of utterances in a flight booking dialogue. It also described the implementation of a small application that uses the model. The trained model achieves considerably high scores (app. 70%) against a pre-annotated corpus, but only 38% of full satisfaction in 1 session with a real user. Although not statistically significant, the user-centred evaluation indicates the obvious but often neglected danger that pre-annotated corpora do **not** cover all/most scenarios that real users expect such models to handle. This makes it imperative that models intended to be used in production be evaluated by real users and re-trained accordingly, or at the very least, that corpora to train models be collected and curated in a way that they are representative of expected interactions with real users. The high performance with the corpus evaluation demonstrate that, given the right quality of a corpus, a high-quality, production-ready model is feasible.