We tested the model on the whole corpus to see how well it could retrieve 10 topics corresponding to the 10 scenarios.

Modules

all_scenarios.py

The module:

- extracts all the stories from the InScript corpus.
- creates the term **dictionary** of the courpus, where every unique term is assigned an index.
- converts a list of the documents into Document Term Matrix (**corpus**) using the dictionary prepared above.
- trains the **LDA model** with 10 topics (using the gensim library) on the Document Term Tatrix.

The dictionary, the corpus and the lda_model we used are saved to file in the current folder.

NOTE: LdaModel gives slightly different results every time we run it. Don't run the module if you want to visualize our results.

all_scenarios_eval.py

The module was used to output 'all_scenarios_results.txt'. It stores:

- a dictionary of counts of how many documents belonging to a specific scenario were assigned to each topic (e.g. if a document originally belonging to the scenario 'bath' is assigned to topic 0, increase dict[topic_0][bath] by 1). A document is assigned to the topic with the highest probability on the document.
- the same dictionary, normalized by number of documents per scenario, to visualize the distribution over topics of the documents belonging to a scenario.
- the 15 words with the highest probability for each topic.

all_scenarios_eval_20.py

Given the corpus and the LDA model trained with 20 topics, prints the proportion of documents captured by the 10 best topics.

all_scenarios_visual.py

Run only this module to see a visualization of the model with 10 topics with pyLDAvis.

Universität des Saarlandes
WiSe 2016/17
Computational Linguistics

Procedure and Scores

The InScript corpus contains 910 stories instantiating 10 specific scenarios (on average 91 stories per scenario): taking a bath, changing a bicycle tire, taking a bus ride, baking a cake, taking a flight, grocery shopping, getting a haircut, going to the library, taking a train trip, planting a tree.
We ran the LDA model with 50 passes, 100 iterations and 10 topics on the whole InScript corpus (1 document = 1 full story) using the all_scenarios.py module. To evaluate the model, we ran the module all_scenarios_eval.py. The results were saved to 'all_scenarios_results_10.txt'.
We here report, in table format, the dictionary of counts of how many documents belonging to a specific scenario were assigned to each topic, normalized by number of documents per scenario. The value highlighted is the highest per scenario:

	Bath	**Bicycle**	**Bus**	**Cake**	**Flight**	**Grocery**	**Haircut**	**Library**	**Train**	**Tree**
0	0.032	0.092	0.022	0.021	1.0	0.116	0.125	0.043	0.425	0.033
1	0.117	0.138	0.217	0.010	0.0	0.158	0.182	0.957	0.092	0.352
2	0.064	0.023	0.0	0.010	0.0	0.011	0.261	0.0	0.057	0.615
3	0.053	0.034	0.141	0.124	0.0	0.716	0.080	0.0	0.0	0.0
4	0.255	0.057	0.0	0.103	0.0	0.0	0.352	0.0	0.0	0.0
5	0.479	0.092	0.022	0.041	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.563	0.185	0.072	0.0	0.0	0.0	0.0	0.011	0.0
7	0.0	0.0	0.011	0.619	0.0	0.0	0.0	0.0	0.046	0.0
8	0.0	0.0	0.011	0.0	0.0	0.0	0.0	0.0	0.172	0.0
9	0.0	0.0	0.391	0.0	0.0	0.0	0.0	0.0	0.195	0.0

The 15 words with the highest probability for each topic:

0
('0.034*"plane" + 0.025*"flight" + 0.016*"seat" + 0.015*"airport" +
0.013*"ticket" + 0.012*"went" + 0.010*"got" + 0.010*"airplane" + 0.009*"bag" +
0.008*"hour" + 0.008*"time" + 0.008*"took" + 0.008*"security" +
0.008*"attendant" + 0.008*"get"')

1
('0.083*"book" + 0.043*"library" + 0.017*"card" + 0.014*"read" + 0.013*"went" +
0.011*"librarian" + 0.010*"one" + 0.010*"found" + 0.010*"back" + 0.009*"find" +
0.009*"check" + 0.008*"wanted" + 0.008*"took" + 0.008*"would" + 0.007*"home"')

2
('0.089*"tree" + 0.030*"hole" + 0.020*"plant" + 0.015*"would" + 0.014*"dirt" +
0.012*"root" + 0.009*"dug" + 0.009*"soil" + 0.009*"water" + 0.009*"shovel" +
0.008*"decided" + 0.008*"around" + 0.008*"grow" + 0.008*"back" + 0.008*"yard"')

Francesca Capuano

```
3
('0.028*"grocery" + 0.025*"store" + 0.021*"item" + 0.020*"cart" +
0.020*"shopping" + 0.019*"list" + 0.013*"went" + 0.012*"put" + 0.011*"bag" +
0.011*"got" + 0.011*"food" + 0.010*"get" + 0.010*"go" + 0.010*"car" +
0.010*"needed"')

4
('0.071*"hair" + 0.029*"cut" + 0.016*"get" + 0.015*"haircut" + 0.014*"chair" +
0.011*"back" + 0.010*"salon" + 0.009*"went" + 0.009*"barber" + 0.009*"wanted" +
0.009*"told" + 0.009*"appointment" + 0.009*"time" + 0.008*"like" +
0.008*"stylist"')

5
('0.053*"water" + 0.039*"bath" + 0.037*"tub" + 0.015*"get" + 0.015*"hot" +
0.013*"put" + 0.012*"towel" + 0.012*"bubble" + 0.011*"drain" + 0.011*"take" +
0.011*"got" + 0.010*"warm" + 0.010*"body" + 0.010*"soap" + 0.010*"clean"')

6
('0.060*"tire" + 0.028*"bike" + 0.020*"tube" + 0.018*"air" + 0.014*"back" +
0.013*"flat" + 0.012*"put" + 0.011*"bicycle" + 0.010*"hole" + 0.009*"pump" +
0.009*"took" + 0.009*"patch" + 0.009*"would" + 0.008*"ride" + 0.007*"got"')

7
('0.062*"cake" + 0.019*"oven" + 0.017*"pan" + 0.016*"ingredient" + 0.013*"put" +
0.011*"baking" + 0.011*"bowl" + 0.010*"mix" + 0.010*"egg" + 0.009*"bake" +
0.009*"make" + 0.009*"batter" + 0.008*"frosting" + 0.007*"recipe" +
0.007*"chocolate"')

8
('0.045*"train" + 0.020*"ticket" + 0.018*"station" + 0.014*"time" + 0.012*"went"
+ 0.008*"trip" + 0.008*"car" + 0.007*"arrived" + 0.007*"would" + 0.006*"took" +
0.006*"day" + 0.006*"chicago" + 0.006*"next" + 0.005*"bought" + 0.005*"u"')

9
('0.059*"bus" + 0.038*"train" + 0.022*"stop" + 0.014*"got" + 0.014*"get" +
0.014*"seat" + 0.011*"ticket" + 0.010*"ride" + 0.010*"take" + 0.009*"station" +
0.009*"time" + 0.009*"driver" + 0.008*"people" + 0.008*"would" +
0.007*"minute"')
```

For a visualization of the model, run only the module all_scenarios_visual.py.

We also trained the model with 20 topics, to see whether 10 topics would still be prevalent (corpus_20, dictionary_20, lda_model_20, all_scenarios_results_20 and doc_scenarios_20 were saved in the current folder).
We used the module all_scenarios_eval_20.py to print the proportion of documents captured by the 10 best topics: 10 out of the 20 topics still capture more than 90% of all documents (0.90659).

Results

Except for the overlap of ‘flight’ and ‘train’, each topic captured the highest portion of documents of a different scenario.
The overlap of ‘flight’ and ‘train’ is justifiable, as the two scenarios share part of the vocabulary (e.g. "seat", "ticket", "went", "bag", "hour", "time", "took"). Moreover, consider that topic 8 still mainly captures documents from the ‘train’ scenario.
The highest portion of documents from a scenario captured by one topic doesn’t always exceed 50% (see bath, bus, haircut and train). But all in all, there is a discrete one-to-one correspondence of topics and scenarios.

NOTE:
Don’t forget that the size of the corpus is limited, and the documents are small crowdsourced stories: the speakers have much freedom and sometimes go off topic (example from the ‘library’ scenario: “While checking out the book the librarian noticed the handgun on my hip and we had a great conversation about the importance of being ready, willing, and able to protect yourself in any situation and in any locality. She said she was grateful to live in a Nation where we had that right and I heartily agreed with her.”)