Since the model might perform better when the documents display a strong preference for a topic (it was at least partially successful in step 1 but not in step 2), we tested it on the corpus of all sentences from the 97 stories in the CAKE scenario, assuming that speakers tend to organize the text with a sequence similar to the one of the steps of a script.
The idea was that, assuming a number of topics equal to the number of steps in a scenario, LDA could namely identify words describing the same step as belonging to the same 'topic'.
In this case, we didn't assume the number of steps to be equal to the number of labels.
Wanzare et al. (2016) crowdsourced the DeScript Corpus, a corpus of event-sequence descriptions (EDSs) for a number of well-known activities (including BAKING A CAKE). 320 English native speakers were asked to write down one EDS for each scenario, namely a minimum of 5 and a maximum of 16 steps which would instantiate each script. Semi-supervised clustering algorithms were employed to group event descriptions into paraphrase sets. Each paraphrase set was manually labeled (e.g. choose_recipe, buy_ingredients, etc. for the scenario BAKING A CAKE). We considered the number of paraphrase set to be more representative of the number of steps to retrieve in the cake scenario (27).

Modules

cake_sentences.py

The module was used to:

- extract all the sentences from all the stories from the CAKE scenario in the InScript corpus.
- train the LDA model on 2 versions of the corpus: a regular version, and a version in which, during the 'cleaning' phase (remove punctuation and stopwords, lemmatization) we concatenated the lemmas with tag they were associated most often with in the InScript annotation. The second version serves visualization purposes: in the pyLDAvis visualization we can more easily recognize whether a topic captures words belonging to the same tag.
  In both cases, we:
- created the term **dictionary** of the courpus, where every unique term is assigned an index.
- converted the list of the documents into Document Term Matrix (**corpus**) using the dictionary prepared above.
- trained the **LDA model** with 20 topics (using the gensim library) on the Document Term Matrix.

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

The dictionary_with_tags, the corpus_with_tags and the lda_model_with_tags 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 exactly our results.

cake_sentences_eval.py

The module was used to output 'cake_sentences_results.txt'. It stores, in order, the 3 tags most often associated with each topic.

cake_sentences_n_best_topics.py

Out of the 27 topic with thich the original model was trained, prints the minimum number of topics that still capture more than 90% of the sentences.
A topic 'catches' a sentence if it's the topic with the highest probability for that sentence.

cake_sentences_visual.py

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

cake_sentences_with_tags_visual.py

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

cake_sentences_sequence.py

The module finds the best sequence of topics according to the model. The best sequence is found by checking, for all topics, whether they mostly precede or follow the other events in most of the documents.
The module outputs 'all_sentences_sequence_results.txt', which stores the 10 sentences with the highest probability per topic, in the order of topics given by the best sequence.

Universität des Saarlandes
WiSe 2016/17
Computational Linguistics

PHASE 1
Procedure and Scores

The InScript corpus contains 97 stories instantiating the scenario ‘baking a cake’.
We ran the LDA model with 50 passes, 100 iterations and 27 topics on the InScript CAKE corpus (1 document = 1 sentence for all sentences in each story) using the cake_sentences.py module.
We first compared the topics with respect to the 37 labels from the InScript annotation, to check if they would mostly correspond to event labels (‘ScrEv_*’). To do this, we ran the module cake_sentences_eval.py. The results were saved to ‘cake_sentences_results.txt’.
We here report the 3 tags most often associated with each topic, ordered from the most frequent:

```
0     ['UnrelEv', 'Evoking', 'ScrPart_decoration']
1     ['UnrelEv', 'ScrPart_ingredients', 'NPart']
2     ['ScrPart_ingredients', 'UnrelEv', 'ScrPart_utensil']
3     ['UnrelEv', 'NPart', 'ScrPart_ingredients']
4     ['ScrPart_ingredients', 'UnrelEv', 'ScrEv_make_dough']
5     ['NPart', 'UnrelEv', 'ScrPart_ingredients']
6     ['UnrelEv', 'NPart', 'ScrPart_cake']
7     ['ScrPart_utensil', 'UnrelEv', 'ScrPart_ingredients']
8     ['UnrelEv', 'NPart', 'ScrPart_beneficiary']
9     ['UnrelEv', 'ScrPart_ingredients', 'ScrPart_utensil']
10    ['NPart', 'UnrelEv', 'ScrPart_ingredients']
11    ['ScrEv_preheat', 'ScrPart_temperature', 'ScrPart_time']
12    ['UnrelEv', 'ScrPart_utensil', 'ScrPart_ingredients']
13    ['UnrelEv', 'ScrPart_ingredients', 'NPart']
14    ['NPart', 'UnrelEv', 'ScrEv_take_out_oven']
15    ['ScrPart_ingredients', 'ScrPart_utensil', 'ScrEv_preheat']
16    ['ScrPart_ingredients', 'ScrPart_cake', 'ScrEv_grease_cake_tin']
17    ['UnrelEv', 'ScrEv_pour_dough', 'ScrPart_dough']
18    ['ScrPart_ingredients', 'UnrelEv', 'NPart']
19    ['UnrelEv', 'ScrPart_utensil', 'ScrEv_take_out_oven']
20    ['UnrelEv', 'NPart', 'ScrPart_cake']
21    ['UnrelEv', 'No_label', 'ScrPart_ingredients']
22    ['UnrelEv', 'ScrEv_preheat', 'ScrPart_baking_instructions']
23    ['UnrelEv', 'ScrPart_ingredients', 'Evoking']
24    ['UnrelEv', 'ScrPart_ingredients', 'NPart']
25    ['UnrelEv', 'ScrPart_ingredients', 'NPart']
26    ['UnrelEv', 'ScrPart_ingredients', 'ScrPart_cake']
```

Francesca Capuano

We here report the distribution of the first best, the second best and third best tags:

0
'ScrPart_utensil': 1
'NPart': 3
'ScrPart_ingredients': 5
'UnrelEv': 17
'ScrEv_preheat': 1

1
'Evoking': 1
'ScrPart_ingredients': 7
'ScrPart_utensil': 3
'NPart': 4
'UnrelEv': 7
'ScrPart_cake': 1
'ScrPart_temperature': 1
'No_label': 1
'ScrEv_preheat': 1
'ScrEv_pour_dough': 1

2
'Evoking': 1
'ScrPart_utensil': 2
'ScrEv_preheat': 1
'ScrPart_decoration': 1
'ScrPart_cake': 3
'ScrEv_take_out_oven': 2
'ScrPart_ingredients': 6
'ScrPart_baking_instructions': 1
'ScrPart_dough': 1
'ScrPart_beneficiary': 1
'NPart': 5
'ScrEv_make_dough': 1
'ScrPart_time': 1
'ScrEv_grease_cake_tin': 1

Results

We were looking at whether 27 topics as retrieved by the LDA model could overlap with the 27 steps in the cake scenario retrieved from Wanzare et al.
We first compared the topics with respect to the 37 labels from the InScript annotation, to check if they would mostly correspond to event labels (‘ScrEv_*’), since the 20 event labels overlap with the 27 steps.
There is no such correspondence: the distribution of the 3 best tags is similar to the one we retrieved when training the model on the whole documents.
Nevertheless, partitioning the corpus into sentences rather than documents caused a slight refinement and increased the diversification of the topics: the first best tags cover 3 labels relevant to the cake scenario (as opposed to 2 in the documents model), the second best tag covers 6 (vs. 7), the third best tag covers 12 (vs. 9), even though we now have less topics.

The 27 topics don't seem to show a correspondence with the script's steps.

PHASE 2
Procedure and Scores

We tried to find a way to optimize the number of topics. In all_scenarios we knew that the topics were, if not exactly 10, a number close to 10: we saw that, by training the model with 20 topics, the 10 best topics would still capture more than 90% of the documents in the corpus.
We checked how many topics out of 27 covered the 90% of the sentences with the module cake_sentences_n_best_topics.py.
Then, we re-ran the model with the new number of topics (19).
The tags are again not very informative:

```
0     ['UnrelEv', 'ScrEv_decorate', 'ScrPart_cake']
1     ['ScrPart_utensil', 'ScrPart_time', 'UnrelEv']
2     ['UnrelEv', 'NPart', 'ScrPart_ingredients']
3     ['UnrelEv', 'ScrPart_ingredients', 'ScrPart_recipe_source']
4     ['UnrelEv', 'ScrPart_ingredients', 'Evoking']
5     ['UnrelEv', 'ScrEv_eat', 'ScrEv_put_cake_oven']
6     ['UnrelEv', 'ScrPart_temperature', 'ScrPart_cook']
7     ['ScrPart_ingredients', 'Evoking', 'UnrelEv']
8     ['UnrelEv', 'ScrPart_ingredients', 'ScrPart_kitchen/location']
9     ['UnrelEv', 'NPart', 'ScrPart_beneficiary']
10    ['UnrelEv', 'ScrPart_ingredients', 'ScrPart_cake']
11    ['UnrelEv', 'NPart', 'ScrPart_ingredients']
12    ['UnrelEv', 'NPart', 'Evoking']
13    ['ScrPart_ingredients', 'ScrPart_utensil', 'ScrPart_cake']
14    ['UnrelEv', 'ScrPart_ingredients', 'ScrPart_utensil']
15    ['ScrPart_ingredients', 'ScrEv_put_cake_oven', 'NPart']
16    ['ScrPart_ingredients', 'UnrelEv', 'ScrEv_get_ingredients']
17    ['UnrelEv', 'NPart', 'Evoking']
18    ['UnrelEv', 'ScrPart_cake', 'ScrPart_utensil']


0
'ScrPart_utensil': 1
'UnrelEv': 14
'ScrPart_ingredients': 4

1
'ScrEv_eat': 1
'Evoking': 1
'UnrelEv': 1
'ScrPart_temperature': 1
'NPart': 5
'ScrPart_cake': 1
'ScrPart_utensil': 1
'ScrPart_time': 1
'ScrEv_put_cake_oven': 1
```

'ScrPart_ingredients': 5
'ScrEv_decorate': 1

2
'ScrPart_utensil': 2
'UnrelEv': 2
'ScrEv_put_cake_oven': 1
'Evoking': 3
'ScrPart_cook': 1
'ScrPart_beneficiary': 1
'ScrPart_kitchen/location': 1
'NPart': 1, 'ScrPart_cake': 3
'ScrPart_recipe_source': 1
'ScrEv_get_ingredients': 1
'ScrPart_ingredients': 2

We then decided to take a different qualitative approach to retrieve the sequence of steps in the script.
We used the module cake_sentences_sequence.py to output the best sequence of topics according to the model. The best sequence is found by checking, for all topics, whether they mostly precede or follow the other topics in most of the documents. In the file 'cake_sentences_sequence_results.txt', you can see the 10 sentences with the highest probability per topic, in the order of topics given by the best sequence. We manually checked whether a consistent sequence of steps was recognizable across the ordered sets of 10 best sentences per topic.
The best sequence of topics is:
[3, 2, 4, 0, 18, 5, 6, 1, 7, 8, 11, 9, 12, 13, 14, 10, 16, 15, 17]

We examined the 10 best sentences per topic and chose the step that they seemed to represent best. In order:

Topic 3 is maybe ascribable to the step 'choose a recipe'.

Topics 2 and 4 are not uniform or ascribable to a specific step.

Topic 0, although with some outliers, overall seems to reflect the combination of the events 'take the cake out of the oven' + 'let the cake cool down'.

Topics 5 and 18 are not uniform or ascribable to a specific step.

Topic 6 overall seems to reflect the step 'preheat the oven'.

Topic 1 contains 3 sentences referring to the event 'take the cake out of the pan'.

Maybe 7 can be loosely linked to 'enjoy the cake'.

Topic 8 is maybe ascribable to 'get ingredients'.

Topic 11 contains 3 sentences about 'check whether the cake is done'.

Topic 9 is maybe ascribable to 'deciding to bake a cake for x's birthday'.

Topic 12 is not uniform or ascribable to a specific step.

Francesca Capuano

Topic 13 is maybe ascribable to ‘mix the ingredients’.

Topic 14 can be loosely linked to ‘smooth the mixture’.

Topic 10 can be loosely linked to ‘follow the recipe’.

Topic 16 is a mixture of ‘gather ingredients’ + ‘mix the ingredients’

Topic 15 is maybe ascribable to ‘grease the pan’.

Topic 17 is not uniform or ascribable to a specific step.

Sometimes the step is only recognizable in 3 or 4 of the 10 sentences (see ‘only loosely linked to’).

Results

No order is recognizable, comparable to the sequence of steps needed to bake a cake.
Only ‘choose ingredients’ as first step, the sequence ‘gather ingredients’ + ‘mix the ingredients’+’grease the pan’ and the sequence ‘mix the ingredients’+’smooth the mixture’ seem to be reasonable.
We couldn’t unravel the internal structure of the CAKE script, but the 10 best sentences per topic sometimes reflect ‘a step in the sequence’. It seems at least that working with sentences rather than topics brought to a partial improvement.
The question is very complex and would require a more in-depth analysis, including a testing of the algorithms used on test cases and gold-standards.