SURV704 – Computer Based Content Analysis II

– Practical Project on Text Mining –

Elisabeth Linek

1. Introduction

The goal of my project is to identify topics and their frequencies within open text entries coming from social science surveys, more concretely being answers to open ended questions.

Due to the fact of not having real data at hand I decided to simulate the task based on book reviews, where amazon users were able to review books with their own words. Based on that I tried to treat these texts as answers to the question "*How would you review the book you just bought?*".

There is a lot of textual data collected based on open-ended questions, since social science exists, there are questions asked without settled answer option. This is at one hand a gift to every researcher, because it is direct feedback – unscaled and with original words of the respondents. On the other hand, the so gathered data is often analysed "by hand", meaning there is a lot of manpower and time included, going through all the texts, cleaning, sorting and classifying the answers, so that the results are tangible.

I will process and analyse the data as open-ended text passages in order to build a routine that is pre-processing, structuring, tokenizing and analysing these texts in terms of showing frequencies.

I am aware of the necessity of a validation of the so gained insights, but the focus of the project will be lying on the text preparation and processing, as it is the most important step in order to get trustworthy or better said reliable results based on textual data from questionnaires.

2. THE DATA

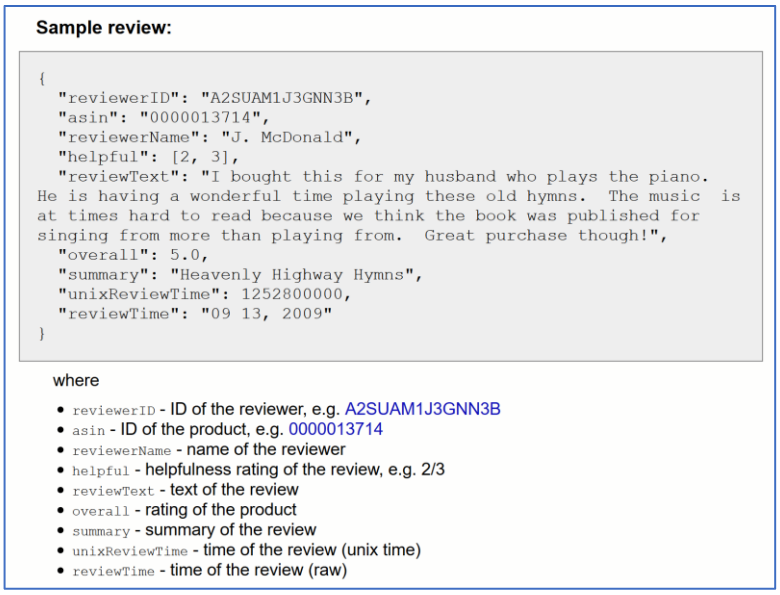
In order to set up the model, I used a large data-file of book reviews from amazon, that was provided for research aspects by McAuley and J. Leskovec, who have conducted themselves a lot of research regarding content analysis.

This dataset contains book reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014, the downloaded file is a selection of book reviews, containing 8,898,041 reviews. Furthermore, I set up the python notebook where the modelling was specified and setup on a sub selection of all available text, I selected one year out of the given years. That restriction was only done due to processing power of my device.

After loading the data, I separated a selection of years in order to save processor workload and started the analysis and modelling by looking at the structure of the data itself in order to setup the pre-processing the best possible way. Once, the corpus was defined, I started with a selection of relevant texts, only taking the review texts into account, in order to focus on the texts that simulate the answers to an open-ended question, leaving additional data from the review aside.[1]

1 The source of data is: http://jmcauley.ucsd.edu/data/amazon/
This dataset contains book reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014, the downloaded file is a selection of book reviews, containing 8,898,041 reviews.

Here is an example how the downloaded reviews are structured:


Sample review:

```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

where

- `reviewerID` - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- `asin` - ID of the product, e.g. 0000013714
- `reviewerName` - name of the reviewer
- `helpful` - helpfulness rating of the review, e.g. 2/3
- `reviewText` - text of the review
- `overall` - rating of the product
- `summary` - summary of the review
- `unixReviewTime` - time of the review (unix time)
- `reviewTime` - time of the review (raw)

As the example shows, there is additional information and metadata included, which I did not consider for the purpose of the here presented project. The focus set here, due to the research goal to simulate "human written answers to open-ended questions", was on the review text itself, tagged as "*reviewText*".

3. THE METHODS

3.1. Pre-Processing the data

The project aims to build an LDA model, that provides the analyst with a matrix of topics based on frequency and probability of word stated together within the analysed texts. Before being able to set up the LDA model, some relevant pre-processing steps were needed to prepare the textual data in order to run the model and present valid results.

Starting the analysis of the text with the following pre-processing steps, presented within the order, that these steps were applied in the script:

Lemmatization → building bigrams → Lowercasing (→ stop word removal) → Tokenization

Lemmatization was applied in order to reducing all the different occurring word-forms to one single, for example, reducing "builds", "building", or "built" would all be reduced to the form of "build". I was applying the lemmatization procedure in order to reduce the complexity f textual data and m let the topics together to a denser corpus, aiming to lead to a better statistical base for the occurrence of relevant word within the later applied LDA modelling.

Furthermore, I created bigrams within the pre-processing, in order to obtain more "related" words, since the LDS model itself is not taking any sentiments into account but rather focuses the frequencies of occurring words.

The later applied stop-word-removal lead in further modelling to some problems, it was difficult to find a level of stop-word-removal without losing to much information or content-related, relevant information, so that I excluded that part of pre-processing for the moment. To define the line of relevant small words compared to rather senseless" small words is a very small line.

Finalizing the pre-processing with the step of tokenizing the so far prepared data frame into a list of words, removing punctuations and unnecessary characters created a corpus that was ready to apply the LDA model for detecting "outstanding" topics within the given data base.

3.2. THE LDA MODEL

Once the corpus was prepared, based on the above described pre-processing procedures, the LDA modelling was set up to be applied on the data.

LDA or latent Dirichlet allocation is a so called "generative probabilistic model" of a corpus of composites made up of parts or tokens. In terms of topic modelling, the composites are documents and the parts are words and or bi-grams, as they were defined within the presented project.

LDA models are comparable to probabilistic latent semantic analysis (pLSA), with the difference, that in LDA models the topic distribution is assumed to have a sparse Dirichlet prior probabilistic distribution. The sparse Dirichlet priors encode the intuition, that documents in general cover a small set of topics and that those topics use only a small set of words frequently. In practice, this results in a better disambiguation of words or tokens and leads to a more precise assignment of documents to topics, which was my reason to decide to base the topic modelling on LDA methods rather than probabilistic latent semantic analysis.

Spoken in the terminology of results from the given project, a rather large set of "related word clouds" – each of them defining a topic itself – was spilled out, with the need to define a title, that suits the word-selection best. A selection of results is presented in the following paragraph.

4. THE RESULTS

The above described procedures of pre-processing and modelling led to a bunch of topics, each defined by a selection of 10 words, related by proportions; a selection of these are presented here:

```
[…]
(40,
 '0.063*"travel" + 0.046*"letter" + 0.043*"meditation" + 0.021*"hotel" + '
 '0.015*"read " + 0.015*"holiday" + 0.015*"island" + 0.014*"defy" + '
 '0.013*"trip" + 0.011*"last,"'),
 (65,
 '0.024*"health" + 0.020*"life" + 0.015*"exercise" + 0.011*"partner" + 0.010*"story" + '
 '0.008*"man" + 0.008*"knowledge" + 0.007*"food" + 0.007*"family" + 0.006*"time"'),
 (16,
 '0.020*"love" + 0.017*"camp" + 0.015*"roll" + 0.015*"genre." + '
 '0.015*"dating" + 0.015*"game." + 0.013*"fast-paced" + 0.012*"feelings" + '
 '0.011*"foster" + 0.011*"romance,"'),
 (13,
 '0.042*"musical" + 0.029*"band" + 0.026*"video" + 0.019*"trail" + '
 '0.013*"adventure." + 0.012*"compile" + 0.011*"fantasy" + 0.010*"suicide" '
 '+ 0.010*"lines." + 0.009*"fire."'),
(56,
 '0.056*"german" + 0.038*"ritual" + 0.031*"hitler" + 0.030*"vote" + '
 '0.030*"nazi" + 0.015*"war" + 0.014*"rape" + 0.012*"write," + '
 '0.012*"allies" + 0.012*"bombing"'),
(34,
 '0.027*"family" + 0.015*"story" + 0.011*"wonderful" + 0.010*"collection." + '
 '0.010*"friends," + 0.010*"book" + 0.008*"pets" + 0.008*"adam" + '
 '0.007*"intro" + 0.007*"scheme"'),
 (10,
 '0.015*"authentic" + 0.015*"christian," + 0.014*"fanatic" + 0.014*"church" + '
 '0.012*"god" + 0.011*"found." + 0.011*"frankly," + 0.011*"cherish" + '
 '0.010*"grim" + 0.009*"hell"'),
 (83,
 '0.032*"my" + 0.026*"lifestyle" + 0.024*"construction" + 0.019*"pleasant" + '
 '0.018*"dramatic" + 0.015*"topic," + 0.014*"par" + 0.013*"comedy" + '
 '0.012*"fine," + 0.012*"mood"'),
[...]
```

Topic 10 for example is a represented as '0.015*"*authentic*" + 0.015*"*christian*," + 0.014*"*fanatic*" + 0.014*"*church*" + "0.012*"*god*" + 0.011*"*found*" + 0.011*"*frankly*" + 0.011*"*cherish*" + "0.010*"*grim*" + 0.009*"*hell*"'). The top 10 keywords, that contribute to this topic are: 'authentic, 'christian, 'fanatic', 'church' etc., the corresponding weight of 'church' on topic 10 for example is 0.014. These weights reflect how important a single keyword is to the topic, where it is related to. Looking at the shown keywords, I would assume topic 10 to represent the topic "***Religion***" or "***Christianity***", even though not all the ten keywords would suit that title perfectly.

As interpretation of the here presented results, based on the corresponding words, I defined the following topics based on the given data:

[...]

Topic 40:	travel letter meditation hotel read holiday island defy trip las	→ holidays or travel
Topic 65:	health life exercise partner story man knowledge food family time	→ health or aspects of healthy living
Topic 16:	love camp roll genre dating game fast-paced feelings foster romance	→ romance
Topic 13:	musical band video trail adventure compile fantasy suicide lines fire	→ music
Topic 56:	german ritual hitler vote nazi war rape write allies bombing	→ second world war
Topic 34:	family story wonderful collection friends book pets adam intro scheme	→ family and friends (rather undefined)
Topic 10:	authentic christian fanatic church god 1found frankly cherish grim hell	→ religion or Christianity
Topic 83:	my lifestyle construction pleasant dramatic topic par comedy fine mood	→ undefined!

[...]

A very positive aspect on the conducted analysis is, that the topics are presented in a neutral way, there is no direct connation added, that would bring a positive or negative classification. In terms of social survey science and of course depending on the data or the question, that the data refers to (from the questionnaire), the topics might reveal keywords that show a positive or negative direction, but it would just be a keyword not necessarily defining the overall topic.

On the other hand, and a very critical aspect I experienced within the project, is, that some topic-clouds are a colourful potpourri of keywords, where the definition or derivation of an overall topic is just impossible. For example, the last topic from the shown examples (topic 83) is rather difficult to classify under one main subject. I must admit, that in such cases it would be helpful to have some kind of "trace back option", in order to see directly some examples of text passages, where these words were clustered in a surrounding area. The modelling itself does not bring a completely satisfactory result for all topics.

But still, to get through all the data, all texts by manpower, it would have taken a lot more time, and in terms of difficulties, it would have been beyond any scale- an enormous project to analyse the data. Would it have let to similar topics? A direct comparison would be interesting, further research on how human beings classify texts and capture contents even on large amounts of data (complex data) compared to the text mining techniques would be very interesting.

As an overview or first attempt to get into large amounts of data, the topic model techniques are a very helpful tool.

Not applied in the context of the here presented project, I would be interested in testing and comparing the results of different LDA model techniques; some research during the data preparation for the project brought for example the so called LDA Mallet Model to my attention, which is said to lead to higher quality of keyword-relations within the topics. To apply both models to one data set, test and compare the results would be an interesting aspect for further research on the project.

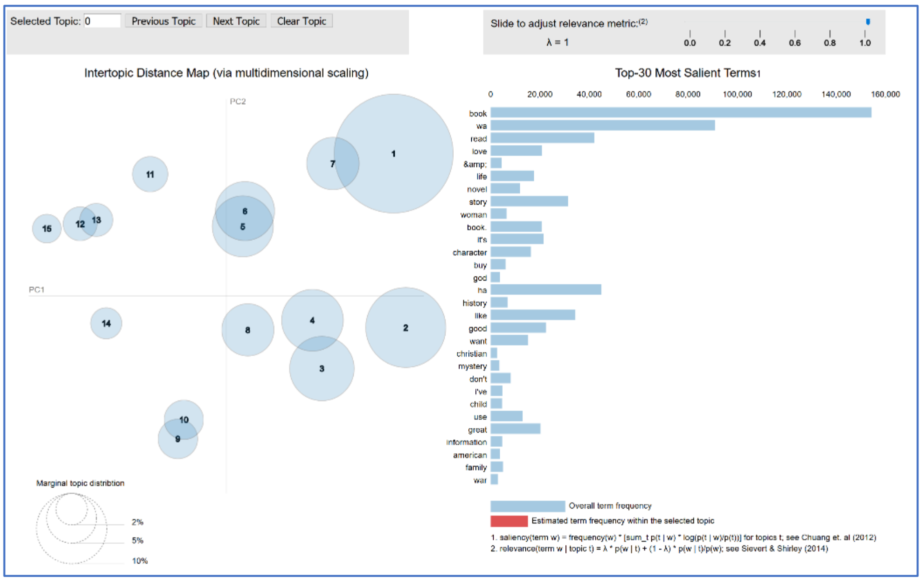
Of course, I se many aspects that could be changed or adjusted within the modelling. As already mentioned, the pre-processing of the data could be adjusted in different directions – pre-processing and "data curation"; the LDA model itself could be set up differently, for example the number of topics could be set to a different amount.

Finally, the quality of the underlying data itself is having impact on the results, that all the modelling brings.

5. THE VISUALIZATION

In order to support the analytical view on the data, a visualization of the results was set up based on the results of the LDA-model. For this purpose, I was working with the pyLDAvis package, building an interactive chart based on the before defined or created topics.

Here is an example, how the interactive chart brings the topics in a relation to each other:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Each "ball" on the plot at the left side represents a topic; the larger the ball is, the more prevalent is the topic based on the textual data that was analysed. A model with a rather valid output is supposed to show fairly large, but non-overlapping balls, scattered throughout the whole chart, instead of being clustered in one of the quadrants or corners. A model with too many topics, will typically have many overlaps, small sized balls, often-times clustered in one area within the chart.

The interactive chart makes it very easy to get into the defined topics. Moving the cursor over the balls or bubbles a data bar-chart on the right part will update and reveal the content of the current topic, which makes the chart a very helpful tool.

Drawing back the here presented visual results, to the before described and discussed results of the LDA model, that the created model does not provide the best possible fit, regarding the size and position within the chart.

6. THE CONCLUSION

What I conclude, based on the results I was able to generate based on the LDA model, is, that some further pre-processing or different aspects of pre-processing of the textual data could have led to some extend to concreter results or different results, leading to different interpretation. While setting up the model I experienced the pre-processing to some extend being a trade-off decision, between preparing the data base for the modelling and- on the other hand- make an impact on the results, meaning to actively participate on the creation of the results, which I see is a very narrow line, when the method is used in terms of data analysis. It almost seems to be a philosophical question: To what extend should the data be touched within the pre-processing without changing its meaning or even in terms of bring up the underlying statement?

The here presented results are not ideal, in terms of relying on these without any further validation. It would have gone beyond the scope of my project, so that I did not set up a validation process for the model, but I would strongly recommend or better said insist on this final step, when using the here presented model based on “real survey data”.

Furthermore, I would conclude, that for the purpose I set up the project, the LDS modelling does not fulfil the tasks perfectly, a sentiment analysis in terms of detecting the underlying meaning of texts could lead to further insights. I may conclude from the conducted project, that further research on the usage of text mining scripts on open ended question analysis will be needed and is probably touching a larger field than expected.

An honest conclusion of the here presented project is, that I experienced the limitations of LDA modelling, in terms of motivation to conduct further practice on the topic of content analysis, topic modelling and above that, sentiment analysis.

APPENDIX

The script, performing the presented results:

```
import logging
from gensim import utils
from gensim import models
from gensim.corpora.textcorpus import TextCorpus, strip_multiple_whitespaces, remove_stopwords
from gensim.models.phrases import Phrases, Phraser
from pprint import pprint

from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet as wn

import pandas as pd
import psutil
import sys
import os

logger = logging.getLogger(__name__)

#json_file = './head.json'
json_file = './2000.json'
#json_file = './reviews_Books_5.json'

bigram_model = None

def memory_usage_psutil():
  # return the memory usage in MB
  import psutil
  process = psutil.Process(os.getpid())
  mem = process.memory_info()[0] / float(2 ** 20)
  return mem

def to_unicode(text, encoding='utf8', errors='strict'):
  """ adapted from TextCorpus.lower_to_unicode (just unicode)
  """
  return utils.to_unicode(text, encoding, errors)

wn_lemm = WordNetLemmatizer()
def lemmatize(tokens):
  return [wn_lemm.lemmatize(token) for token in tokens]

def lemmatize2(tokens):
  r = []

  for token in tokens:
    lem = wn.morphy(token)
    if lem:
      r.append(lem)
    else:
      r.append(token)

  return r

def apply_bigrams(tokens):
  if bigram_model is None:
    logger.error("no bigram model")

  return bigram_model[tokens]

def lowercase(tokens):
  """Remove stopwords using list from `gensim.parsing.preprocessing.STOPWORDS`.
  """
  return [token.lower() for token in tokens]

def split_tokenizer(text):
  for token in text.split():
    yield token

# not yet: bigrams and lemmatization
class CorpusTest(TextCorpus):
  #stopwords = set('for a of the and to in on'.split())
  filename = json_file
```

```
    def get_texts(self):
      reader = pd.read_json(self.filename, lines=True, chunksize=1)
      for chunk in reader:
        # more input modifications?
        yield self.preprocess_text(chunk.reviewText.values[0])
        #yield utils.to_unicode(chunk.reviewText.values[0]).split()

      #for doc in self.getstream():
      #  yield [word for word in utils.to_unicode(doc).lower().split() if word not in self.stopwords]

    def __len__(self):
      self.length = sum(1 for _ in self.get_texts())
      return self.length

    def gen_bigram_model(self):
      bigrams = Phrases(iter(self.get_texts()))
      print(memory_usage_psutil())
      return Phraser(bigrams)

  corpus = CorpusTest(input="not_used",
           character_filters=[to_unicode, strip_multiple_whitespaces],
           token_filters=[lemmatize2, lowercase, remove_stopwords],
           tokenizer=split_tokenizer)

  # create bigram model
  bigram_model = corpus.gen_bigram_model()

  bicorpus = CorpusTest(input="not_used",
            character_filters=[to_unicode, strip_multiple_whitespaces],
            token_filters=[lemmatize2, lowercase, remove_stopwords, apply_bigrams],
            tokenizer=split_tokenizer)

  lda_model = models.LdaModel(bicorpus, id2word=bicorpus.dictionary, num_topics=100)

  pprint(lda_model.print_topics())
  doc_lda = lda_model[corpus]

  print("done")
```

#### Trying to visualize the topic models:

```
  import pyLDAvis
  import pyLDAvis.gensim
  import matplotlib.pyplot as plt
  %matplotlib inline

  import logging
  logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.ERROR)

  import warnings
  warnings.filterwarnings("ignore",category=DeprecationWarning)

  # Compute Perplexity
  #print('\nPerplexity: ', lda_model.log_perplexity(corpus))

  # Computation of Coherence Score
  #coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dictionary=bicorpus.dictionary, coherence='c_v')
  #coherence_lda = coherence_model_lda.get_coherence()
  #print('\nCoherence Score: ', coherence_lda)

  # Visualize the topics
  #pyLDAvis.enable_notebook()
  #vis = pyLDAvis.gensim.prepare(lda_model, bicorpus, bicorpus.dictionary)
  #vis

  #saving the vis.:

  vis = pyLDAvis.gensim.prepare(lda_model, bicorpus, bicorpus.dictionary)
  pyLDAvis.save_html(vis, "test.html")

  print("done too")
```