University of Arkansas at Little Rock

Information Science, Principles and Theory.

Information Science

Mallet vs GenSim: Topic modeling for 20 news groups report.

Authors:
Islam. Akef
Juan S. Muñoz Arango

Advisor:
Dr. Xiaowei Xu

April 23, 2016

1 Introduction

On this resport we are going to work with *Mallet*[1] and *GenSim*[2] and compare them to see how good or bad both work on the topic modelling task. We also want to see how data should be prepared for each tool and see how easy is to get each app up and running.

For both reports we are going to use as an input data the 20 newsgroups data set[3]; This data set contains data from different topics like hardware, computers, motorcycles, politics, electronics, religion among others. The idea of this project is to train a topic model with all the datasets together, experiment with stop words, number of topics and then analyze the extracted topics and display the top 20 words that have the highest probabilities for each topic.

Latent Dirichlet Allocation (LDA) is a topic model that generates topics based on word frequency from a set of documents. LDA is particularly useful for finding reasonably accurate mixtures of topics within a given document set.

We will use Latent Dirichlet Allocation to perform topic modeling and discover semantic structure of the provided dataset, by examining word statistical co-occurrence patterns within the provided dataset titled 20 Newsgroups which consists of 20000 messages taken from 20 newsgroups. The messages are provided by the School of Computing at Carnegie Mellon University which dates back to 1999. The 20000 messages represent 1000 use net articles with approximately 4% of the articles cross posted, they are typical postings thus they have headers with subject lines, signature files and quoted portions of other articles, also each newsgroup is stored in a subdirectory, with each article stored as a separate file.

This dataset comes organized by date, and also comes with some noise (like the "From", "Subject" headers on each post as mentioned before and some typos in the dataset aswell); some of the topics are very closely related to each other like Hardware MAC and Hardware PC, while other topics are very dissimilar like christian topics and motorcycles.

The unsupervised approach provided with LDA allow expressing the documents in the new semantic representation and queried for topical similarity against other documents, this approach is based mainly on creating a term document matrix through performing singular value decomposition, and identifying the occurrences of a certain terms and assigning weights.

Some of the advantages of LDA can be summarized by the following: simple model based on linear algebra, term weights not binary, allows computing a continuous degree of similarity between queries and documents, allows ranking documents according to their possible relevance, Allows partial matching.

The limitations of LDA includes: long documents are poorly represented because they have poor similarity values, search keywords must precisely match document terms; word substrings might result in a "false positive match", semantic sensitivity; documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match", the order in which the terms appear in the document is lost in the vector space representation, theoretically assumes terms are statistically independent and last weighting is intuitive but not very formal.

After getting the results we plan to compare both *Mallet* and *GenSim* against each other and see how different or similar both topics end up being trained.

2 Approaches and Methodologies

On this section we are going to discuss how we installed Mallet and GenSim applications and what procedures we did in order to get to our conslusions, we also are going to mention issues we found and how we solved them.

2.1 *Mallet: Machine Learning for Language Toolkit*

Mallet is a a Java-based console applicaction for language processing, document classification, clustering, topic modeling, etc. It uses Latent Dirichlet Allocation, Pachinko Allocation and Hierarchical Latent Dirichlet Allocation for topic modelling.

In order to setup *Mallet*, one has to download and install the Java SDK and later on download the binaries for Mallet. On this report we are using *mallet-2.0.8RC3* version and we followed the guidelines proposed by Shawn Graham, Scott Weingart and Ian Milligan [4]. These guidelines are quite straight forward and helped us out make Mallet work in around an hour.

Still one has to be carefull to call any Mallet binary from exactly "mallet/", because if you execute the binary from inside "mallet/bin/" it will look like it works but when you run the data it complains with java exceptions. In other words:

Example on how to NOT run the app as it will throw exceptions

```
Console$:./mallet train-topics --input Test20Newsgroups.mallet
Error: Could not find or load main class cc.mallet.topics.tui.TopicTrainer
```

Example on how to properly run mallet

```
Console$:./bin/mallet train-topics --input Test20Newsgroups.mallet
```

2.1.1 Mallet workflow

In order to make Mallet work for topic modelling, there are two steps that need to be followed; the topic importing from the raw data set and the topic training.

A)Topic importing: This is the most important step, in this part we tell mallet what is the data that is going to be used. In this step we pre-process the data and clean it of noise as this is the data that is going to be used for training purposes. After this step a "*.mallet*" binary gets generated with the raw data extracted, processed and free of noise for the next step.

B)Topic training: On this part we use the pre-processed output on the *topic importing* step as an input and we train our topics based on the data we got on the first step. Hence, if the importing process on step 1 still has noise, it will negatively affect the results for the trained data. On this step we can tweak how the training process is done. We can tell mallet the number of topics we want to have, the number of iterations we want to achieve while training the data, set parameters like the beta value for smoothing doc-topic distributions among other values.

Finally after following this seemengly simple 2 steps we get the results in 3 files. The first one contains the topic key words, the second one contains a matrix that contains the most prominent documents for each topic and the third one contains a Gibbs sampling state after each of the iterations. This sampling tells us the words, which topic each word belongs to, the source of each word and the alpha and beta values for the trained model.

2.2 *GenSim: Topic modeling for humans*

Gensim is a python package based on numpy and scipy packages. Genism features are: memory independence where there is no need for the whole training corpus to be on the RAM at any given time, Gensim introduces Simplistic implementations for tf-idf and Latent Semantic Analysis and Latent Dirichlet Allocation, also provides the ability to write simple similarity queries for documents in their semantic representation, Gensim is based mainly on the concepts of a corpus, vector, models and sparse matrices.

Going through the definitions of these concepts: a corpus is a whole dataset provided in a compressed or non-compressed single file path, a vector is a set of defining attributes for the corpus represented in word by word vector value, a model is the algorithm used to extract and find pattern in the document matrix represented by the sparse matrices.

2.2.1 GenSim workflow

The ways to process documents are so varied and application- and language-dependent, a document is represented by the features extracted from it, not by its string form, using the bag of words technique. In this representation, each document is represented by one vector where each vector element represents a question-answer pair.

There exist several file formats for serializing a Vector Space to disk. Gensim implements them via its streaming corpus interface: Documents are read from disk in a lazy fashion, one document at a time, without the whole corpus being read into main memory at once. Gensim transform documents from one vector representation into another. This process serves two goals: To bring out hidden structure in the corpus, discover relationships between words and use them to describe the documents in a new and more semantic way, also to make the document representation more compact and this will lead to improving efficiency and efficacy.

Transformations always convert between two specific vector spaces. The same vector space must be used for training as well as for subsequent vector transformations. Failure to use the same input feature space, such as applying a different string preprocessing, using different feature ids, or using bag-of-words input vectors where tf-idf vectors are expected, will result in feature mismatch during transformation calls.

3 Experiments and Evaluation

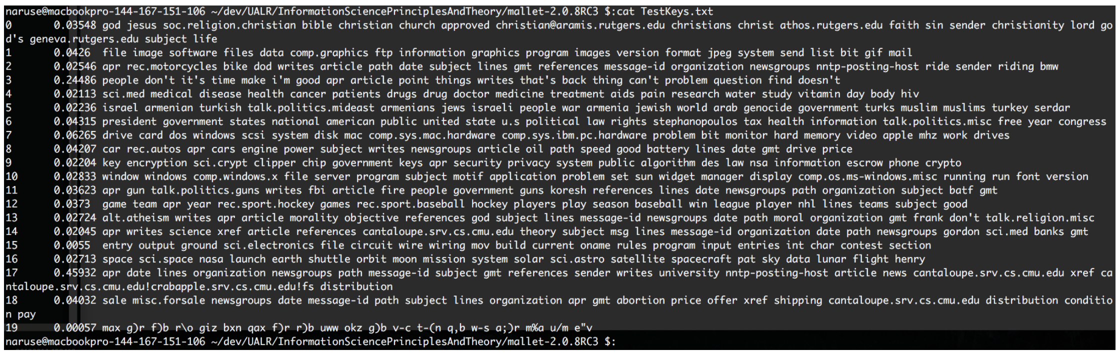
3.1 *Mallet:*

In order to import the data we can tell Mallet what part of the data can be excluded either by a stop-word list, a created stop word list or a regular expression that gets run against the data before processing it.

The initial command we attemted to run was:

```
$Console$:./bin/mallet import-dir --input ../20_newsgroups/
--output Test20Newsgroups.mallet --remove-stopwords  --keep-sequence
```

Where we just tell it to import the whole Directory of the data set *(20_newsgroups)* and remove the stop words found in the english language. Afterwards, we just run the training command to get the keys (Figure: 1).



```
naruse@macbookpro-144-167-151-106 ~/dev/UALR/InformationSciencePrinciplesAndTheory/mallet-2.0.8RC3 $:cat TestKeys.txt
0       0.03548 god jesus soc.religion.christian bible christian church approved christian@aramis.rutgers.edu christians christ athos.rutgers.edu faith sin sender christianity lord go
d's geneva.rutgers.edu subject life
1       0.0426  file image software files data comp.graphics ftp information graphics program images version format jpeg system send list bit gif mail
2       0.02546 apr rec.motorcycles bike dod writes article path date subject lines gmt references message-id organization newsgroups nntp-posting-host ride sender riding bmw
3       0.24486 people don't it's time make i'm good apr article point things writes that's back thing can't problem question find doesn't
4       0.02113 sci.med medical disease health cancer patients drugs drug doctor medicine treatment aids pain research water study vitamin day body hiv
5       0.02236 israel armenian turkish talk.politics.mideast armenians jews israeli people war armenia jewish world arab genocide government turks muslim muslims turkey serdar
6       0.04315 president government states national american public united state u.s political law rights stephanopoulos tax health information talk.politics.misc free year congress
7       0.06265 drive card dos windows scsi system disk mac comp.sys.mac.hardware comp.sys.ibm.pc.hardware problem bit monitor hard memory video apple mhz work drives
8       0.04207 car rec.autos apr cars engine power subject writes newsgroups article oil path speed good battery lines date gmt drive price
9       0.02204 key encryption sci.crypt clipper chip government keys apr security privacy system public algorithm des law nsa information escrow phone crypto
10      0.02833 window windows comp.windows.x file server program subject motif application problem set sun widget manager display comp.os.ms-windows.misc running run font version
11      0.03623 apr gun talk.politics.guns writes fbi article fire people government guns koresh references lines date newsgroups path organization subject batf gmt
12      0.0373  game team apr year rec.sport.hockey games rec.sport.baseball hockey players play season baseball win league player nhl lines teams subject good
13      0.02724 alt.atheism writes apr article morality objective references god subject lines message-id newsgroups date path moral organization gmt frank don't talk.religion.misc
14      0.02045 apr writes science xref article references cantaloupe.srv.cs.cmu.edu theory subject msg lines message-id organization date path newsgroups gordon sci.med banks gmt
15      0.0055  entry output ground sci.electronics file circuit wire wiring mov build current oname rules program input entries int char contest section
16      0.02713 space sci.space nasa launch earth shuttle orbit moon mission system solar sci.astro satellite spacecraft pat sky data lunar flight henry
17      0.45932 apr date lines organization newsgroups path message-id subject gmt references sender writes university nntp-posting-host article news cantaloupe.srv.cs.cmu.edu xref ca
ntaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs distribution
18      0.04032 sale misc.forsale newsgroups date message-id path subject lines organization apr gmt abortion price offer xref shipping cantaloupe.srv.cs.cmu.edu distribution conditio
n pay
19      0.00057 max g)r f)b r\o giz bxn qax f)r r)b uww okz g)b v-c t-(n q,b w-s a;)r m%a u/m e"v
naruse@macbookpro-144-167-151-106 ~/dev/UALR/InformationSciencePrinciplesAndTheory/mallet-2.0.8RC3 $:
```

Figure 1: Initial topic keys. LogLikelihood: -9.45506

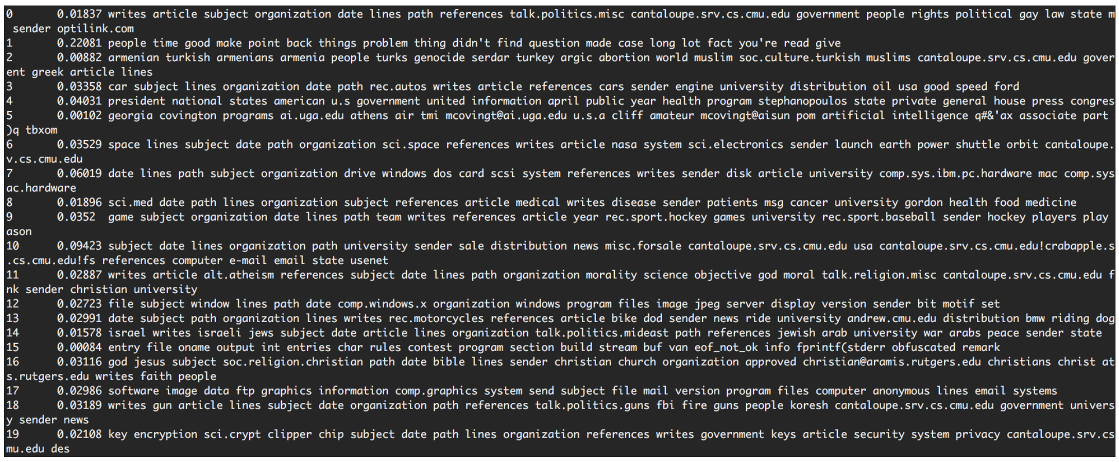
As noted in Figure 1 none of the topics make sense, there are even words that dont make sense at all, like “apr” (Topic 17) or garbage words like the ones found in topic 19.

After finding these issues we decided that we needed an extra hand made stop-words list for this specific data set. Enter the “–extra-stopwords” command!.

We iterated on this extra stop words file several times until we got something that showed words that made sense. We ended up with these stop words:

```
apr max g)r f)b r\o giz bxn qax f)r r)b uww okz g)b v-c t-(n q,b
w-s a;)r m%a u/m e"v xref batf message-id newsgroups rutgers i'm
it's i've don't can't doesn't x-newsreader mailing list
nntp-posting-host reply-to gmt that's t)s g)%a f"z qtm t]f v?%q
p\w u%w tg%q lg)r
```

We ran again the training model with the extra stop words added and we got some better results.



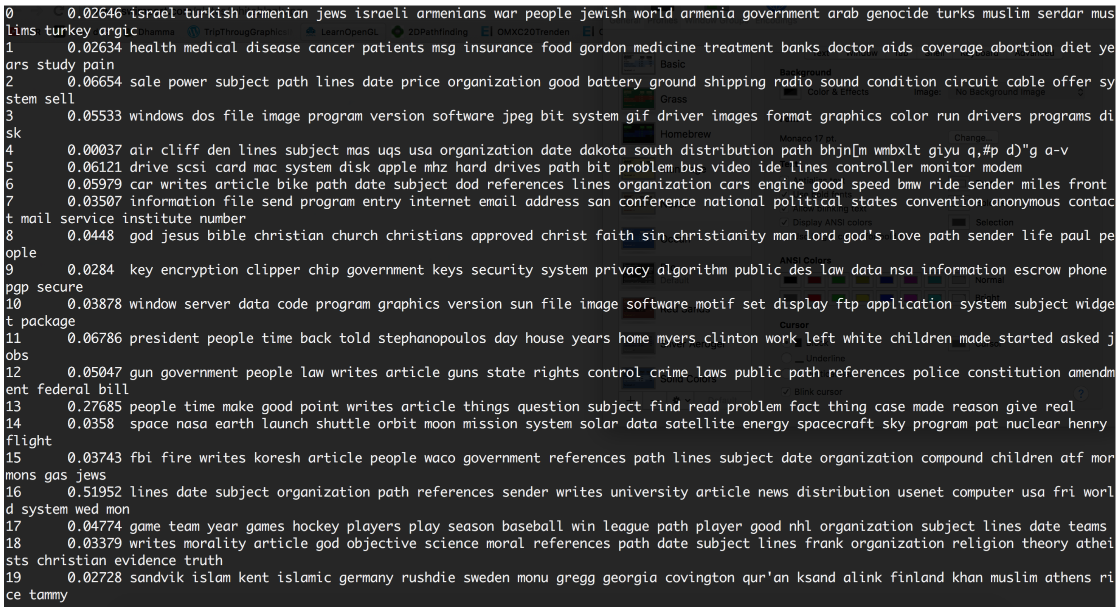
```
0       0.01837 writes article subject organization date lines path references talk.politics.misc cantaloupe.srv.cs.cmu.edu government people rights political gay law state m
 sender optilink.com
1       0.22081 people time good make point back things problem thing didn't find question made case long lot fact you're read give
2       0.00882 armenian turkish armenians armenia people turks genocide serdar turkey argic abortion world muslim soc.culture.turkish muslims cantaloupe.srv.cs.cmu.edu gover
ent greek article lines
3       0.03358 car subject lines organization date path rec.autos writes article references cars sender engine university distribution oil usa good speed ford
4       0.04031 president national states american u.s government united information april public year health program stephanopoulos state private general house press congres
5       0.00102 georgia covington programs ai.uga.edu athens air tmi mcovingt@ai.uga.edu u.s.a cliff amateur mcovingt@aisun pom artificial intelligence q#&'ax associate part
)q tbxom
6       0.03529 space lines subject date path organization sci.space references writes article nasa system sci.electronics sender launch earth power shuttle orbit cantaloupe.
v.cs.cmu.edu
7       0.06019 date lines path subject organization drive windows dos card scsi system references writes sender disk article university comp.sys.ibm.pc.hardware mac comp.sys
ac.hardware
8       0.01896 sci.med date path lines organization subject references article medical writes disease sender patients msg cancer university gordon health food medicine
9       0.0352  game subject organization date lines path team writes references article year rec.sport.hockey games university rec.sport.baseball sender hockey players play
ason
10      0.09423 subject date lines organization path university sender sale distribution news misc.forsale cantaloupe.srv.cs.cmu.edu usa cantaloupe.srv.cs.cmu.edu!crabapple.s
.cs.cmu.edu!fs references computer e-mail email state usenet
11      0.02887 writes article alt.atheism references subject date lines path organization morality science objective god moral talk.religion.misc cantaloupe.srv.cs.cmu.edu f
nk sender christian university
12      0.02723 file subject window lines path date comp.windows.x organization windows program files image jpeg server display version sender bit motif set
13      0.02991 date subject path organization lines writes rec.motorcycles references article bike dod sender news ride university andrew.cmu.edu distribution bmw riding dog
14      0.01578 israel writes israeli jews subject date article lines organization talk.politics.mideast path references jewish arab university war arabs peace sender state
15      0.00084 entry file oname output int entries char rules contest program section build stream buf van eof_not_ok info fprintf(stderr obfuscated remark
16      0.03116 god jesus subject soc.religion.christian path date bible lines sender christian church organization approved christian@aramis.rutgers.edu christians christ at
s.rutgers.edu writes faith people
17      0.02986 software image data ftp graphics information comp.graphics system send subject file mail version program files computer anonymous lines email systems
18      0.03189 writes gun article lines subject date organization path references talk.politics.guns fbi fire guns people koresh cantaloupe.srv.cs.cmu.edu government univers
y sender news
19      0.02108 key encryption sci.crypt clipper chip subject date path lines organization references writes government keys article security system privacy cantaloupe.srv.cs
mu.edu des
```

Figure 2: Initial topic keys with hand made stop word list. LogLikelihood: -9.20506

Figure 2 Looks better, but still we have some words that doesnt tell us anything, i.e “*sci.crypt*”. So for this case, we noticed that all these words could be took away with a simple regular expression, so we did that and created a small regular expression that filters all the words that have points in between.

```
RegExp: (?!^[^.]+$).+
```

The results, much much better. Now we have single words that have a meaning and topics make sense. Figure 3.



```
0       0.02646 israel turkish armenian jews israeli armenians war people jewish world armenia government arab genocide turks muslim serdar mus
lims turkey argic
1       0.02634 health medical disease cancer patients msg insurance food gordon medicine treatment banks doctor aids coverage abortion diet ye
ars study pain
2       0.06654 sale power subject path lines date price organization good battery ground shipping radio sound condition circuit cable offer sy
stem sell
3       0.05533 windows dos file image program version software jpeg bit system gif driver images format graphics color run drivers programs di
sk
4       0.00037 air cliff den lines subject mas uqs usa organization date dakota south distribution path bhjn[m wmbxlt giyu q,#p d)"g a-v
5       0.06121 drive scsi card mac system disk apple mhz hard drives path bit problem bus video ide lines controller monitor modem
6       0.05979 car writes article bike path date subject dod references lines organization cars engine good speed bmw ride sender miles front
7       0.03507 information file send program entry internet email address san conference national political states convention anonymous contac
t mail service institute number
8       0.0448  god jesus bible christian church christians approved christ faith sin christianity man lord god's love path sender life paul pe
ople
9       0.0284  key encryption clipper chip government keys security system privacy algorithm public des law data nsa information escrow phone
pgp secure
10      0.03878 window server data code program graphics version sun file image software motif set display ftp application system subject widge
t package
11      0.06786 president people time back told stephanopoulos day house years home myers clinton work left white children made started asked j
obs
12      0.05047 gun government people law writes article guns state rights control crime laws public path references police constitution amendm
ent federal bill
13      0.27685 people time make good point writes article things question subject find read problem fact thing case made reason give real
14      0.0358  space nasa earth launch shuttle orbit moon mission system solar data satellite energy spacecraft sky program pat nuclear henry
flight
15      0.03743 fbi fire writes koresh article people waco government references path lines subject date organization compound children atf mor
mons gas jews
16      0.51952 lines date subject organization path references sender writes university article news distribution usenet computer usa fri worl
d system wed mon
17      0.04774 game team year games hockey players play season baseball win league path player good nhl organization subject lines date teams
18      0.03379 writes morality article god objective science moral references path date subject lines frank organization religion theory athei
sts christian evidence truth
19      0.02728 sandvik islam kent islamic germany rushdie sweden monu gregg georgia covington qur'an ksand alink finland khan muslim athens ri
ce tammy
```

Figure 3: Topic keys, hand made stop words list and Regex. LogLikelihood: -9.10242

After this we started to play with the simulation values. we changed the Beta and the Alpha values and started to tweak the number of topics. After all of this we played also with the number of iterations and we noticed that the value that made most change in the log likelihood was the number of topics. We managed to reduce the log likelihood to *-8.77896* with 500 topics.

Some examples of the results with Mallet include:

```
Topic 195: printer print font fonts postscript printing laser
printers deskjet color windows dpi ink driver canon truetype
lines paper characters memory

Topic 322: image jpeg images gif bit file color format display
colors quality version formats programs convert free software
viewer conversion tiff

Topic 412: political rights party states human parties state
vote minority public candidates equal law freedom present
democratic covenant law respect civil
```

3.2 *GenSim:*

As mentioned before; data in GenSim gets feeded in a lazy way, meaning one file at the time; this is for avoiding the need to hardware with high RAM. In GenSim we divided the process in 3 parts: *Data cleaning*, *constructing the document matrix* and finally *applying the LDA model to* *the data.*

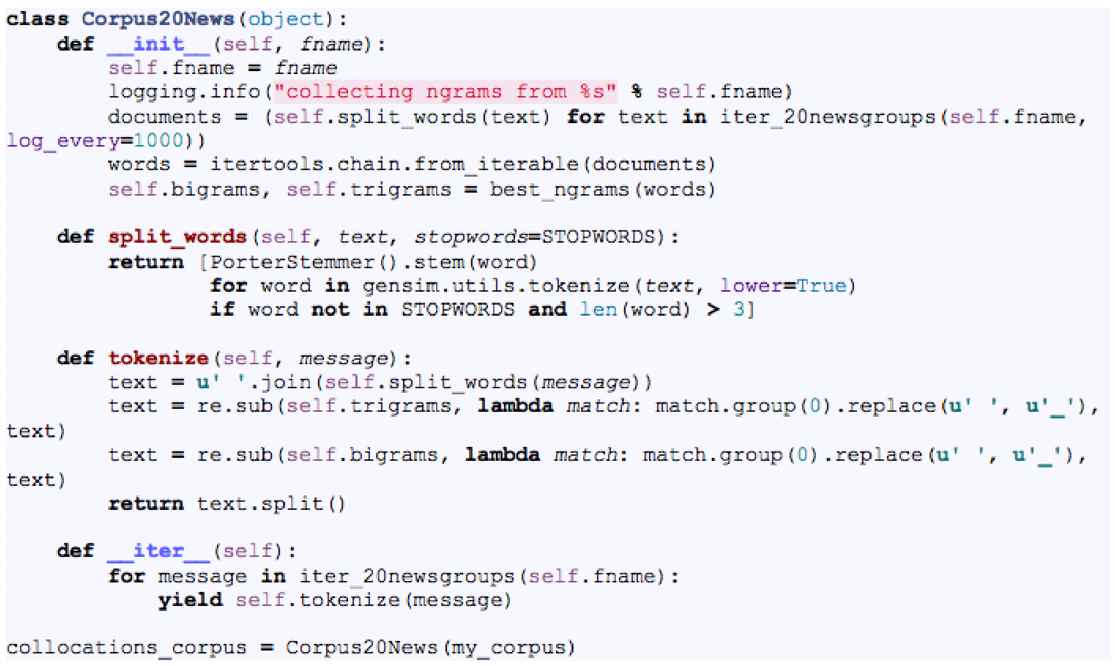
3.2.1 Data cleaning

Data cleaning is crucial for generating a useful topic model the following steps are common to most natural language and text processing methods: Tokenizing, Stopping and Stemming.

Tokenization segments a document into its atomic elements. In this case, we are interested in tokenizing to words. Certain parts of English speech, like conjunctions (for, or) or the word the are meaningless to a topic model. These terms are called stop words and need to be removed from our token list.

Stemming words reduces those terms to stem. This is important for topic modeling, which would otherwise view those terms as separate entities and reduce their importance in the model.

For stemming the data set we create an object that has the processed corpus (Figure: 4).



```
class Corpus20News(object):
    def __init__(self, fname):
        self.fname = fname
        logging.info("collecting ngrams from %s" % self.fname)
        documents = (self.split_words(text) for text in iter_20newsgroups(self.fname,
log_every=1000))
        words = itertools.chain.from_iterable(documents)
        self.bigrams, self.trigrams = best_ngrams(words)

    def split_words(self, text, stopwords=STOPWORDS):
        return [PorterStemmer().stem(word)
                for word in gensim.utils.tokenize(text, lower=True)
                if word not in STOPWORDS and len(word) > 3]

    def tokenize(self, message):
        text = u' '.join(self.split_words(message))
        text = re.sub(self.trigrams, lambda match: match.group(0).replace(u' ', u'_'),
text)
        text = re.sub(self.bigrams, lambda match: match.group(0).replace(u' ', u'_'),
text)
        return text.split()

    def __iter__(self):
        for message in iter_20newsgroups(self.fname):
            yield self.tokenize(message)

collocations_corpus = Corpus20News(my_corpus)
```

Figure 4: Code snippet that creates an object of the whole corpus.

In this process we follow these steps:

1) Remove header and footer from the dataset and return the trimmed string of the file content

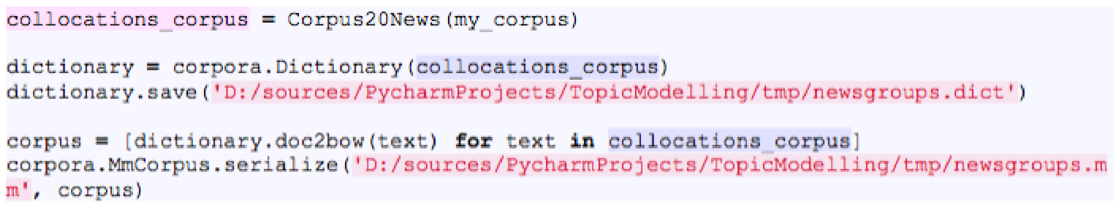
2) Split the text into tokens and apply stemming

3) Remove stop words from the text

3.2.2 Constructing the document term matrix.

The result of our cleaning stage is that texts are tokenized, stop words free and stemmed list of words from a whole corpus. But before applying the models we need to save our corpus on disk as a binary file to be processed later and apply the model on.

We do this in Figure **??** by saving the processed corpus to disk in *GenSim* in two files: *Document matrix* and a *Dictionary*. To generate an LDA model, we need to understand how frequently each term occurs within each document.

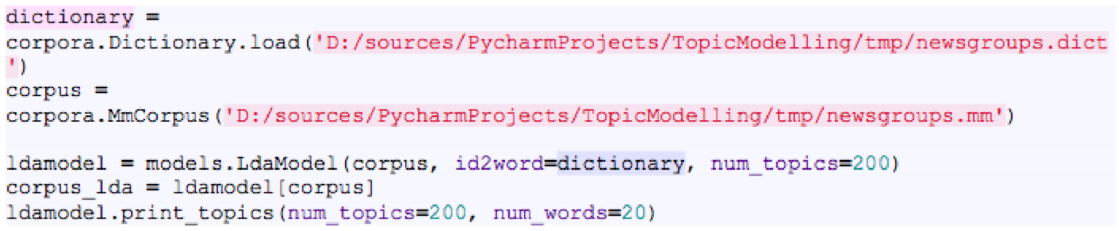


```
collocations_corpus = Corpus20News(my_corpus)

dictionary = corpora.Dictionary(collocations_corpus)
dictionary.save('D:/sources/PycharmProjects/TopicModelling/tmp/newsgroups.dict')

corpus = [dictionary.doc2bow(text) for text in collocations_corpus]
corpora.MmCorpus.serialize('D:/sources/PycharmProjects/TopicModelling/tmp/newsgroups.m
m', corpus)
```

Figure 5: Code snippet that generateds a document term matrix from the processed corpus

3.2.3 Applying the LDA model

Finally, we load the saved dictionary and document matrix. We then apply the LDA model on the processed corpus and run the script shown in Figure 6.



```
dictionary =
corpora.Dictionary.load('D:/sources/PycharmProjects/TopicModelling/tmp/newsgroups.dict
')
corpus =
corpora.MmCorpus('D:/sources/PycharmProjects/TopicModelling/tmp/newsgroups.mm')

ldamodel = models.LdaModel(corpus, id2word=dictionary, num_topics=200)
corpus_lda = ldamodel[corpus]
ldamodel.print_topics(num_topics=200, num_words=20)
```

Figure 6: Code snippet that applies the LDA model and prints results

Our LDA model is now stored as ldamodel. We will run the training model on 500 topics then we can review our topics with the print_topic and print_topics methods, the output below shows 3 topics in within each topic are the 20 most probable words to appear in that topic. Even though our document set is small the model is reasonable.

```
Topic 161: 0.153*government + 0.111*president + 0.030*food +
0.022*protection + 0.022*western + 0.020*governments +
0.020*british + 0.019*countries + 0.014*health_care +
0.014*leaders + 0.014*democratic + 0.012*population +
0.011*democracy + 0.011*policies + 0.011*people + 0.010*today +
0.010*social + 0.008*steve + 0.008*workers + 0.007*involve
```

```
Topic 253: 0.149*arab + 0.063*moon + 0.046*paragraph + 0.032*sins +
0.028*stars + 0.027*appear + 0.017*operation + 0.015*length +
0.013*veal + 0.012*origin + 0.012*entity + 0.012*purpose +
0.011*saving + 0.007*beasts + 0.007*slavery + 0.007*standpoint +
0.007*mathematical + 0.007*symbolic + 0.006*congregation +
0.006*observed

Topic 6: 0.054*russian + 0.047*jesus + 0.041*heaven + 0.032*hockey +
0.031*peter + 0.026*village + 0.020*lord + 0.019*said + 0.019*unto +
0.015*rose + 0.015*spoke + 0.014*judas + 0.013*disciples +
0.012*john + 0.011*battle + 0.010*matthew + 0.009*came +
0.008*saints + 0.008*puck + 0.007*crew
```

4 Conclusions

LDA assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution. Gensim is a python library with good tools to work flexibly on topic modeling, however genism does not provide an out of the box running commands to perform topic modeling, it requires python knowledge.

Mallet is an out of the box tool but unfortunately it doesnt let you tweak or see in between steps like GenSim does. Mallet regardless being a console application is much more user friendly than GenSim, but for advanced work is better to use GenSim as it lets you tweak more parameters than Mallet

Data cleanness is critical, it is extremely important to check the data that we are going to work with before training it, this is because as we exposed, the very beggining attemps throwed random words.

Mallet and GenSim results are a little bit dissimilar but we think this is because of the data we used, even though it came from exactly the same place we dont know which stop words Mallet uses *vs* GenSim and we dont know how the cleanning process gets treated inside each of the tools.

The words that we got out of each tool some of the words had correlation but some other topics didnt. We think this could be enhanced by tweaking more internal parameters in each of the tools.

References

[1] McCallum, Andrew Kachites. 2002. Mallet: MAchine Learning for Language Toolkit. [ONLINE] Available at: http://mallet.cs.umass.edu/. [Accessed 20 April 2016].

[2] Radim Rehurek. 2009. GenSim. Topic Modelling for humans. [ONLINE] Available at: https://radimrehurek.com/gensim/. [Accessed 20 April 2016].

[3] jrennie. 2008. 20 Newsgroups data set. [ONLINE] Available at: http://qwone.com/ jason/20Newsgroups/. [Accessed 20 April 2016].

[4] Shawn Graham, Scott Weingart and Ian Milligan. 2012. Getting Started with topic modelling and MALLET. [ONLINE] Available at: http://programminghistorian.org/lessons/topic-modeling-and-mallet. [Accessed 20 April 2016]

[5] Matthew L Jockers. *Text Analysis with R for students of Literature* Springer, June 10 2014, 145-147

[6] Vector space model, Wikipedia. [Online]. Available at: https://en.wikipedia.org/wiki/vector_space_model. [Accessed: 16 April 2016].

[7] Latent Dirichlet Allocation (LDA) with Python, Latent Dirichlet Allocation (LDA) with Python. [Online]. Available at: https://rstudio-pubs-static.s3.amazonaws.com/79360_850b2a69980c4488b1db95987a24867a.html. [Accessed: 22 April 2016].

[8] Topic Modeling for Fun and Profit, Notebook. [Online]. Available at: http://radimrehurek.com/topic_modeling_tutorial/2%20-%20topic%20modeling.html [Accessed: 22 April 2016].