

CS909 Week 10: Text classification, clustering and topic models

Samuel McDermott u1466355

April 22, 2015

1 Introduction

This report demonstrates the use of text classification, clustering and topic models for the Reuters-21578 dataset [1]. This dataset consists of 21578 documents, extracted from the Reuters newswire in 1987, each with multiple or no labels. The aim of this work is to test a variety of features (topic models, n -grams), as well as text classifiers (naive Bayes, Support Vector Machine, Maximum Entropy, Neural Networks, and Random Forests) and clustering (K-means, Hierarchical Clustering, Expectation Maximisation).

The work in this project was done using R and several packages (cited as used). The code associated with this report can be found at <http://git.io/vvNci>.

2 Preprocessing and Data Cleaning

2.1 Text preprocessing

Associated R code: `TextPreprocessing.R`

The Reuters-21578[1] dataset is a `.csv` file which consists of the label for document, the title of the article and the text in the article.

Some articles have several labels, and some have none. The first step is to take this information apart, so that in the final dataset, each document has only one label. This means that the same document may appear several times in the corpus, once for each label. This was done to ensure that each label accurately contained each document associated with it.

The next stage is to select the 10 most popular labels in the dataset. These were provided to us and are: *earn*, *acquisitions*, *money-fx*, *grain*, *crude*, *trade*, *interest*, *ship*, *wheat*, *corn*. This reduces the dataset size and provides a more concentrated selection of documents to label. The documents are then randomly ordered, so that k-fold evaluation can be carried out later.

At this stage, we now have 9612 documents, with the 10 most popular labels, their titles, and their manually added labels. This array can be passed into the other functions, `lda`, `featureClassification`, `featureClustering`

2.2 Document Term Matrix

Associated R code: `convertToDtm.R`

A document term matrix (DTM) is a matrix that describes the frequency of terms that occur in a collection of documents. The rows correspond to the documents in the collection, and the columns correspond to the terms.

In the first stage of each text analytical function, a DTM is calculated for the inputted array. This makes the code more reusable, as a user only has to enter the label, title and text into each of the functions. This is achieved using the `tm` package [2].

2.2.1 Corpus

Firstly, the documents to be converted into a DTM are changed into a Corpus. This is just a character vector for each document, with a few attributes used by the package. Some more preprocessing is achieved with the function `tm_map`:

- **tolower**: This turns all the data into lowercase, so that uppercase letters (for example at the beginning of sentences) are ignored.
- **removeWords, stopwords("english")**: Stopwords are words that are filtered out to improve the performance of natural language processing. These tend to be the most common words in a language, which do not provide much information gain. These stopwords are provided from the Snowball stemmer project[3] in English.
- **removePunctuation**: This removes non alpha-numeric characters from the documents as they are are unneeded and provide little information gain.
- **stemDocument**: To stem a word is to reduce it to its word stem, base or root form. For example: *argue*, *argued*, *argues*, *arguing* all reduce to the stem *argu*. This is done for all the words in the document using Porter's stemming algorithm[4] and Snowball[3]. Stemming is done as it is useful to make connections between derivationally related words as well as reduce sparseness.
- **removeNumbers**: This removes numeric characters from the documents as they are unneeded and provide little information gain.
- **stripWhitespace**: There may be extra whitespace in the documents, either from the input, or from the preceding transformations. Multiple whitespace characters are therefore collapsed into a single blank.
- **PlainTextDocument**: Mainly for correcting formatting errors.

The documents are now in an informationally useful format and can be turned into a DTM. The next stage is to create the word features for each document.

2.2.2 *n*-grams

I am using *n*-grams (also known as *Shingles*) for my document features. A *n*-gram is a contiguous sequence of *n* items (in this case words) from the document. This is an extension of a *bag-of-words* technique, which can be represented as a 1-gram. Intuitively, these *n*-length phrases can reduce the uncertainty of the meaning of single words and provide more context. In this report, I will test 1,2 and 3-grams. This is achieved using the `RWeka` package[5].

The resulting corpus can now be convert into a DTM, ready for topic models, classification and clustering. Only words that are longer than three characters are included to reduce redundant information. In addition, I found it useful to reduce the sparsity of the DTM to 0.98, to reduce the size of the resulting DTM for processing whilst not losing too much information. This resulting DTM had 1047 terms and 9612 documents for unigram features. `tf-idf` or *term frequency-inverse document frequency* was used as the weighting. This value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the whole corpus. This helps to adjust for words appearing more frequently in general.

3 Topic models

Associated R code: `lda.R`

Topic models are algorithms that find hidden thematic structure in documnet collections. Given that a document is about a particular topic, we should expect to see certain words appearing in it more frequently. This algorithm finds these 'most relevant' words for each unsupervised topic.

Latent Dirichlet Allocation (LDA)[6] is a type of topic model where each topic is assumed to be characterised by a particular set of topics, similar to the standard *bag-of-words* model. Having created a DTM as described previously, I applied an LDA model using the `topicmodels` package[7]. Using *uni-grams*, and finding 10 topics, the following table shows the top 10 words associated with each topic:

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
said	said	said	trade	said	dlrs	mln	cts	cts	billion
oil	bank	company	tonnes	will	share	reuter	net	april	pct
prices	rate	inc	wheat	offer	quarter	total	loss	record	year
will	rates	shares	said	agreement	year	interest	mln	reuter	last
government	dollar	reuter	exports	reuter	per	end	shr	pay	said
production	market	corp	grain	new	earnings	tax	profit	prior	rose
reuter	pct	pct	imports	board	sales	assets	revs	dividend	february
also	banks	stock	japan	per	first	year	reuter	march	january
industry	exchange	group	reuter	may	company	compared	note	may	expected
last	interest	share	export	spokesman	operations	four	avg	div	compared

It is possible to see how these terms might be connected:

- Topic 1 appears to be connected to *crude*.
- Topic 2 appears to be connected to *money-fx*.
- Topic 3 does not immediately appear to have a connection.
- Topic 4 appears to be connected to *grain* or *wheat*.
- Topic 5 may be connected to *acquisitions*.
- Topic 6 does not immediately appear to have a connection.
- Topic 7 does not immediately appear to have a connection.
- Topic 8 does not immediately appear to have a connection.
- Topic 9 does not immediately appear to have a connection.
- Topic 10 may be connected to *earn*.

However, it is not always immediately obvious, especially as some of the topics are very similar.

We can repeat this for *bi-grams*:

Topic 1 mln stg vs mln vs billion billion vs mln vs money market bank england billion stg profit mln england said	Topic 2 company said inc said per share corp said crude oil dlrs per common stock also said dlrs share oil prices	Topic 3 mln tonnes department said us agriculture soviet union said reuter agriculture department securities exchange exchange commission last month sources said	Topic 4 united states interest rates central bank analysts said billion dlrs officials said dealers said last year sources said foreign exchange	Topic 5 cts vs cts prior qty div vs cts div cts record april prior pay record march april reuter march reuter
Topic 6 mln vs vs profit vs mln net loss vs loss shr loss revs mln loss dlrs loss cts loss mln	Topic 7 mln dlrs billion dlrs first quarter last year dlrs mln dlrs reuter mln dlr year ago year earlier dlrs billion	Topic 8 oper net oper shr cts share dlrs cts cts per per share cts oper gain dlrs vs dlrs net excludes	Topic 9 mln vs vs mln cts vs net vs vs cts shr cts cts net revs mln vs revs avg shrs	Topic 10 vs loss cts vs net profit cts net revs vs loss revs shr profit profit vs loss cts avg shrs

This does not appear to make the identities of the topics any clearer. **Topic 2** appears to be related to oil; **Topic 3** appears to be related to grain and agriculture and **Topic 4** appears to be related to money-fx. The other topics are very vague.

Working this time with *tri-grams*:

Topic 1 vs mln avg mln avg shrs avg shrs vs mln vs mln shrs mln vs avg shrs mln vs avg shrs shrs vs note shrs vs reuter revs vs avg	Topic 2 cts per share mln dlrs mln st qtr net mln dlrs reuter dlrs mln dlrs billion dlrs billion sales mln dlrs mln dlrs year net profit mln mln dlrs cash	Topic 3 mln vs mln vs mln note revs mln vs vs mln year mln year shr mln note net mln revs mln barrels per day note net includes vs mln reuter	Topic 4 mln vs mln net mln vs dlrs vs dlrs shr dlrs vs billion vs billion vs dlrs net vs mln revs vs mln nine mln nine mths cts net mln	Topic 5 cts vs cts net vs sales sales mln vs div cts vs vs cts prior vs sales mln cts prior pay qty div cts pay april record record march reuter
Topic 6 cts vs profit vs profit revs net loss vs loss vs profit vs profit cts cts net loss profit cts net profit revs mln shr loss cts loss cts vs	Topic 7 net profit vs profit vs loss cts oper net cts net profit cts vs loss vs loss revs profit cts vs shr profit cts oper net vs vs loss cts	Topic 8 dlrs per share securities exchange commission us agriculture department dlrs cts share mln dlrs cts bank england said uk money market mln dlrs dlrs agriculture department said filing securities exchange	Topic 9 shr cts vs cts net vs cts vs cts vs cts net net vs revs vs revs mln revs mln vs mln vs mln vs mln reuter vs revs vs	Topic 10 net loss vs loss vs loss cts vs loss cts net loss shr loss cts loss cts vs vs loss revs vs loss cts vs loss dlrs loss cts net

Again, this doesn't seem to make the topics any more obvious. We can assume that **Topic 3** is related to oil, and **Topic 8** is related to agriculture, but the rest of the topics are too similar to draw any conclusions.

From this we can see the uni-grams are the best text feature for topic models, although as many of

the topics in the documents are similar, the use of topic models is not particularly useful in this case.

4 Text Classification

Associated R code: `featureClassification.R`

Using unigrams, bigrams and trigrams as my features, I then carried out classification using 5 classifiers: naive Bayes, Support Vector Machine, Maximum Entropy, Neural Networks, and Random Forests.

I created code that could apply as many different classification algorithms for comparison, allowed n -grams for any n , used k -fold classification and produced aggregated analysis.

After creating the DTM as described earlier, using both the title and document texts as documents to create the best dataset, and a user defined n -gram, I then used k -fold selection to maximise the data available for testing. Having already randomised the documents, I slid a testing window across the data and trained the models on the rest.

4.1 Classifiers

The user is free to select which model to use. A very useful package for this is `RTextTools`[8]. This encapsulates many other packages and classification algorithms, providing a uniform interface. It does not, however, incorporate Naive Bayes, and so I added this myself. I have tested 5 classifiers for comparison:

- **Naive Bayes:** This classifier applies Bayes' theorem with strong independence between the features. In this project I use the package `e1071`[9]. This was used as it is a useful benchmark for comparing other classifiers. Call with "NB".
- **Support Vector Machine:** This classifier constructs a hyperplane for use in classification. As a low memory, but previously successful algorithm in the area of text, it should provide good performance. I used a linear kernel as it appears text documents are linearly separable[10]. The package used for this is `e1071`[9]. Call with "SVM".
- **Maximum Entropy:** This is a tool for low memory multinomial logistic regression, implemented by `maxent`[11]. It is probabilistic like Naive Bayes, although does not assume conditional independence between the features. This is particularly true for text classification, where words in a document are obviously not independent. Call with "MAXENT".
- **Neural Networks:** A neural network text classifier is a network of units, where the input units represent terms, the output units represent the labels and the weights on the edges between the units represent dependence relations. This is a higher memory algorithm, and subsequently takes more time than the previous classifiers. However, they have been shown to demonstrate statistically comparable performance to that of other on-line linear classifiers[12]. This implementation uses the package `nnet`[13] and is a feed-forward neural network with a single hidden layer. Call with "NNET".
- **Random Forests:** This algorithm constructs a number of decision trees at training time and outputs the class that is the mode of the classes at testing time. Although this is therefore a high memory algorithm, it shares advantages with *bagging* and corrects for decision trees overfitting tendencies. The package `randomForest`[14] is used. Call with "RF".

4.2 Results and Analysis

The following sections will demonstrate the relative successes of these algorithms. The precision, accuracy and recall are calculated both at the macro and micro level for 10-fold classification. Micro-averaging gives weight to every document classification decision, whereas macro-averaging gives equal weight to each topic. As some of the topics (earn \approx 3900 documents; acquisitions \approx 1830 documents) have considerable more documents than others (wheat \approx 280 documents, corn \approx 240 documents) the macro average is a more realistic indicator of effectiveness.

4.3 Uni-gram results

4.3.1 Naive Bayes

avg-type	precision	accuracy	recall
macro	0.246	0.811	0.128
micro	0.057	0.811	0.895

4.3.2 Support Vector Machine

avg-type	precision	accuracy	recall
macro	0.528	0.952	0.508
micro	0.760	0.952	0.973

4.3.3 Maximum Entropy

avg-type	precision	accuracy	recall
macro	0.453	0.936	0.447
micro	0.680	0.936	0.964

4.3.4 Neural Networks

avg-type	precision	accuracy	recall
macro	0.226	0.928	0.287
micro	0.638	0.928	0.960

4.3.5 Random Forests

avg-type	precision	accuracy	recall
macro	0.522	0.948	0.486
micro	0.739	0.948	0.971

4.4 Uni-gram analysis

It can be seen that the SVM is the most effective algorithm. It has the highest accuracy (0.953), precision (0.528) and recall (0.508) compared to the others. All the algorithms perform better than Naive Bayes, which shows they are all at least as good as the baseline.

4.5 Bi-gram results

4.5.1 Naive Bayes

avg-type	precision	accuracy	recall
macro	0.094	0.813	0.106
micro	0.066	0.813	0.896

4.5.2 Support Vector Machine

avg-type	precision	accuracy	recall
macro	0.484	0.896	0.204
micro	0.481	0.896	0.942

4.5.3 Maximum Entropy

avg-type	precision	accuracy	recall
macro	0.439	0.862	0.182
micro	0.312	0.862	0.924

4.5.4 Neural Networks

avg-type	precision	accuracy	recall
macro	0.161	0.892	0.171
micro	0.461	0.892	0.940

4.5.5 Random Forests

avg-type	precision	accuracy	recall
macro	0.463	0.896	0.200
micro	0.478	0.896	0.942

4.6 Bi-gram analysis

Once again, the SVM algorithm performed best. It has the same accuracy (0.896) as Random Forests, but is more precise (0.484) and has a better recall (0.204). The worst performing algorithm is Naive Bayes, as expected. Bi-grams appear to be less accurate than Uni-grams.

4.7 Tri-gram results

4.7.1 Naive Bayes

avg-type	precision	accuracy	recall
macro	0.072	0.814	0.143
micro	0.070	0.814	0.897

4.7.2 Support Vector Machine

avg-type	precision	accuracy	recall
macro	0.164	0.875	0.163
micro	0.373	0.875	0.930

4.7.3 Maximum Entropy

avg-type	precision	accuracy	recall
macro	0.302	0.889	0.184
micro	0.447	0.889	0.939

4.7.4 Neural Networks

avg-type	precision	accuracy	recall
macro	0.038	0.845	0.097
micro	0.224	0.845	0.914

4.7.5 Random Forests

avg-type	precision	accuracy	recall
macro	0.042	0.845	0.100
micro	0.224	0.845	0.914

4.8 Tri-gram analysis

We can see that the Maximum Entropy algorithm performed best with tri-grams. It also has a considerably higher precision than a SVM. Once again, as expected, the Naive Bayes algorithm performed the worst. Tri-grams are less accurate than bi-grams or uni-grams and so we can conclude the uni-grams are the most effective word feature, and the SVM the best classifier for these text documents.

5 Clustering

This section focusses on clustering algorithms. The same dataset is used, with 10 topics, and I will test the clustering algorithms ability to cluster these documents into 10 clusters, and then compare the results. As it has been previously demonstrated to be the most effective feature, both for topic models and classification, I shall use uni-gram features. The three clustering algorithms I shall test are Hierarchical Agglomerative, k-means and Expectation Maximisation.

5.1 Clustering Algorithms

The three algorithms have been chosen to represent different clustering techniques.

- **k-means:** This algorithm aims to partition n observations into k clusters, such that each observation belongs to the cluster with the nearest mean. As the data has 10 labels, I shall split the data into 10 clusters. This implementation uses the built in `kmeans` function and plotted using the package `cluster`.
- **Hierarchical Clustering:** This algorithm builds a hierarchy of clusters. Each document is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, until there is a single cluster. Then, we place a cut in the result dendrogram at the position where there are 10 clusters. This algorithm is very slow. The implementation used is the standard function `hclust`.
- **Expectation Maximisation:** The EM algorithm iteratively tries to find the parameters of the probability distribution that has the maximum likelihood of its attributes. This implementation uses the package `mclust`[15].

5.2 Results and Analysis

5.2.1 k-means

5.2.2 Hierarchical Clustering

5.2.3 Expectation Maximisation

5.2.4 Rand index

For each clustering algorithm, the Rand index is calculated compared to the manual labels. This is implemented by the `fpc` package[16]. This is a measure of similarity between two data clusterings and has a value between 0 and 1. 0 indicates that the two data clusters do not agree on any pair of points and 1 indicates the data clusters are exactly the same.

References

- [1] D. Lewis. (2013) Reuters-21578, distribution 1.0. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection>
- [2] I. Feinerer and K. Hornik, *tm: Text Mining Package*, 2014, r package version 0.6. [Online]. Available: <http://CRAN.R-project.org/package=tm>
- [3] (2015) Snowball stemmer project. [Online]. Available: <http://svn.tartarus.org/snowball/trunk/website/algorithms/english/stop.txt>
- [4] M. F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.
- [5] K. Hornik, C. Buchta, and A. Zeileis, “Open-source machine learning: R meets Weka,” *Computational Statistics*, vol. 24, no. 2, pp. 225–232, 2009.
- [6] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

- [7] B. Grün and K. Hornik, “topicmodels: An R package for fitting topic models,” *Journal of Statistical Software*, vol. 40, no. 13, pp. 1–30, 2011. [Online]. Available: <http://www.jstatsoft.org/v40/i13/>
- [8] T. P. Jurka, L. Collingwood, A. E. Boydstun, E. Grossman, and W. van Atteveldt, “Rtexttools: Automatic text classification via supervised learning.” [Online]. Available: <http://CRAN.R-project.org/package=RTextTools>
- [9] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C.-C. Chang, and C.-C. Lin, “Misc functions of the department of statistics (e1071), tu wien.” [Online]. Available: <http://cran.r-project.org/web/packages/e1071/index.html>
- [10] T. Joachims, *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.
- [11] T. P. Jurka and Y. Tsuruoka, “maxent: Low-memory multinomial logistic regression with support for text classification.” [Online]. Available: <http://cran.r-project.org/web/packages/maxent/index.html>
- [12] F. Sebastiani, “Machine learning in automated text categorization,” *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.
- [13] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, 4th ed. New York: Springer, 2002, iSBN 0-387-95457-0. [Online]. Available: <http://www.stats.ox.ac.uk/pub/MASS4>
- [14] A. Liaw and M. Wiener, “Classification and regression by randomforest,” *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: <http://CRAN.R-project.org/doc/Rnews/>
- [15] C. Fraley, A. E. Raftery, T. B. Murphy, and L. Scrucca, *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation*, 2012.
- [16] C. Hennig, *fpc: Flexible procedures for clustering*, 2014. [Online]. Available: <http://cran.r-project.org/web/packages/fpc/>