

EC999: Naive Bayes Classifiers (II)

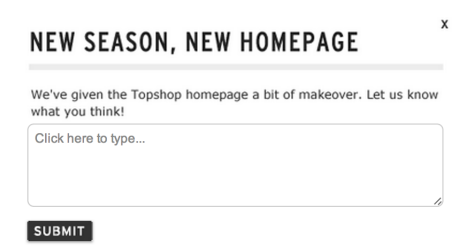
Thiemo Fetzer

University of Chicago & University of Warwick

December 4, 2017

Sentiment Analysis as a use case

- ▶ We illustrate this with a standard example for sentiment analysis; exit feedback surveys.

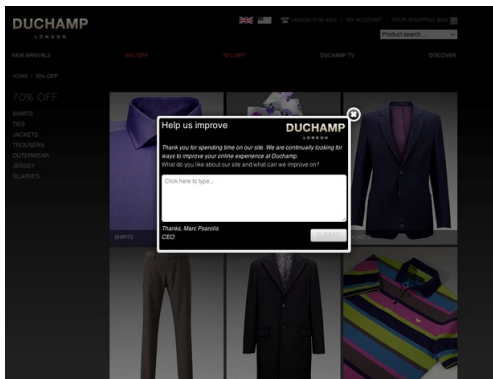


A screenshot of a web form for a feedback survey. The form has a title "NEW SEASON, NEW HOMEPAGE" in bold black text, followed by a horizontal line. Below the line is the text "We've given the Topshop homepage a bit of makeover. Let us know what you think!". Underneath is a large text input field with the placeholder text "Click here to type...". At the bottom of the form is a dark grey button with the word "SUBMIT" in white capital letters. A small "x" icon is in the top right corner of the form's container.

- ▶ Often times, you want to target your attention to negative (but not profane) feedback, as the criticism is what contains information on how to improve.

Sentiment Analysis as a use case

- We illustrate this with a standard example for sentiment analysis; exit feedback surveys.



- Often times, you want to target your attention to negative (but not profane) feedback, as the criticism is what contains information on how to improve.

Some sample exit feedback

- [1] "its all pretty good. cant think of any improvements - good site"
 - [2] "nothing-the site is excellent and easy to use"
 - [3] "Great Service, very speedy delivery, beautiful necklaces - thank you very much"
 - [4] "Actually cant think of any improvements that you could make. Sight is clear, easy to understand and quick.
- Thanks!"
- [5] "Nothing site is easy to use and descriptions of products are very clear"
 - [6] "31215"
 - [7] "Cant think of anything you can improve on - lovely website with gorgeous gifts and easy to navigate."
 - [8] "you are an excellent website and can not think of anything that needs to be improved, its easy to negotiate and order from. have recommended you to many friends."
 - [9] "Wow this website is amazing i love absolutly everything on it. The layout is fantastic and the items you are selling look so lovely i could buy it all. I dont think there is anything you could improve on, brillian 10/10 *****"
 - [10] "nothing very happy so far with all products"
 - [11] "i think the site is wonderful as it is! easy to use, search engine, cleare photographs! FANTASTIC!!"
 - [12] "everything looks great so far, good search options, easy to use too"
 - [13] "I am happy with the web site its clean image is very easy to use"
 - [14] "nothing, I think selection of goods is excellent and website very efficient"
 - [15] "I think it all looks very good"
 - [16] "The functions on the website dont work very well when using Safari and the search bar is not very good."
 - [17] "No problems navigating the site, everything really easy to find. Keep up the good work."
 - [18] "I think notonthehighstreet is great. It has a wide variety of lovely products coevering all tastes and budgets. Great service and the sellers are so easy to deal with. Keep up the good work!!!"
 - [19] "i think its a pretty good site and am impressed."
 - [20] "I think it is a very easy website to use - very happy with it!"
 - [21] "i think there is nothing at all that this website needs to improve on as it is just great!! :D"
 - [22] "very good site"
 - [23] "Nothing so far, thought site very user friendly"
 - [24] "Nothing! I think everything you sell is unique, the website is impecable and extremely easy to use and I admire the fact that you are supporting small businesses in a very corporate high street market! Thank you and please keep it up!"
 - [25] "it actually looks very good to be honest. I like the slides and the layout, good job!"
 - [26] "clunky web site, keeps freezing and very slow!"

Some sample exit feedback

- [1] "I wanted to buy an item - had to register, wait for an email and then begin the process again! what a waste of time! why not select item - register then continue shopping?"
- [2] "do not return to the home page when clicking continue shopping having added something to your basket. Would be better to go back to the page you were on!"
- [3] "yep when doing a search, then being able to put them in price order would be great, eg heart bracelet. also your zoom doesnt work properly."
- [4] "Dont keep bringing this box up every time I click on a page!"
- [5] "I want to give a positive rating for some purchases, where do I do that?"
- [6] "it would be nice to have a clock which would help a child to learn to tell the time ie. with past and to marked and colour coded."
- [7] "dont give me this pop up box when I am trying to make an order!!!"
- [8] "After a search and adding an item to your basket, you hit the continue shopping button it takes you back to the home page and not to your search - grrrr!"
- [9] "I need a gift for an 18 year old girl and would love to be able to click on a button that gives me those options."
- [10] "when clicking on an item from a search result. Make it easier to go back to the same spot on the results page."
- [11] "discount days...where an item is discounted for one day only, giving your customers an opportunity to scoop a deal and return much more frequently! I WOULD hehehe"
- [12] "You could make it cheaper, suitable for every pocket. You could also do a design your own room type thing where you can design a room (ie bedroom/living room etc) using your products. xoxo"
- [13] "give people the right adress on things when ask i need the exact adress on how to get an catalog sent to my home .com"
- [14] "There should be a way that you can save your favourite sellers so you can easily look them up when you return to the website."
- [15] "not having a lightbox pop up box asking for feedback on your index page. Maybe put it after Ive ordered something or had a chance to take a look at your website."
- [16] "would love it if I could store up the sellers I have previously brought from on a sort of my favourite sellers page"
- [17] "you still show things that are out of stock for a long time. Either indicate when they are coming back to stock or take them off the site. thanks."
- [18] "You could get this silly help us improve... box out of my face when im trying to order for a start! :)"
- [19] "its sometimes frustrating to have the picture zoom out when your mouse is hovering on an item. Better to click on the picture to get more details...thanks"

Plan

Text as Data

Naive Bayes illustration, validation, learning curve

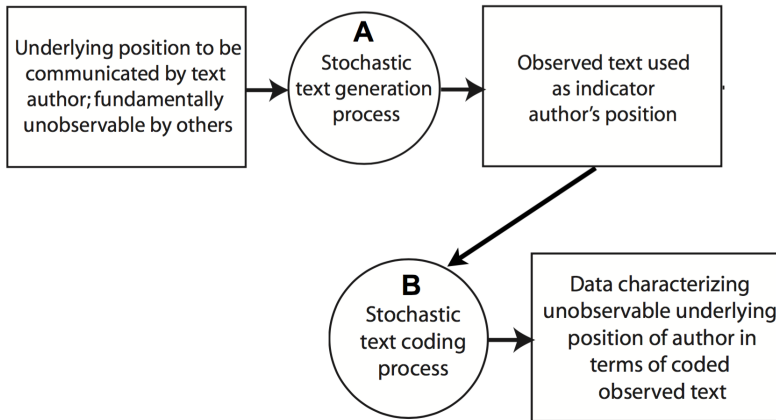
Text as Data

- ▶ Text is a huge and valuable data resource, that is currently still underexploited in the quantitative social sciences.
- ▶ In this section, we will introduce some terminology around treating Text as Data and we will then apply our Naive Bayes classifier to text data.
- ▶ This is where Naive Bayes can really shine.

Key feature of quantitative text analysis

1. *Selecting and conversion*: Defining a corpus, making it machine readable and defining unit of analysis
2. *Defining and selecting features*: These can take a variety of forms, including tokens, equivalence classes of tokens (dictionaries), selected phrases, human-coded segments (of possibly variable length), linguistic features, and more.
3. Conversion of textual features into a *quantitative matrix*
4. A quantitative or statistical procedure to extract information from the quantitative matrix

A stochastic generative view of text data



Two Different Generative Models Explored

We have talked about the Naive Bayes classifier in the context of a simple example, we now apply or adapt this framework to the context of text data. We will work with two different *generative* models for textual data.

1. Bernoulli document model
2. Multinomial document model

From Text to a Quantitative Matrix

Indian security forces shot another militant in Kupwara district. A group of 100 armed Naxalites attacked Asarali police station, about 250 km from the district headquarters of Gadchiroli, on Maharashtra's border with Andhra Pradesh early on Monday. The Vanni Commander of the Sri Lankan Army (SLA), at a meeting at the Vavuniya Headquarters, asked Tamil paramilitary groups to surrender any firearms in their possession before , tomorrow morning. As per the cease-fire agreement reached between the two, , all Tamil paramilitary groups would have either surrender their arms or join the SLA as regulars to be posted in areas away from the North and the East. According to the prosecution, , , Lahori and his associates killed Ramzan Ali and injured two others in the Saddar area of Karachi.

From Text to a Quantitative Matrix



From Text to a Quantitative Matrix

terms	docs				
	1	2	3	4	5
another	1	0	0	0	0
district	1	1	0	0	0
forces	1	0	0	0	0
in	1	0	1	1	1
indian	1	0	0	0	0
kupwara	1	0	0	0	0
militant	1	0	0	0	0
security	1	0	0	0	0
shot	1	0	0	0	0
a	0	1	1	1	1
about	0	1	0	0	0
andhra	0	1	0	0	0
armed	0	1	0	0	0
asarali	0	1	0	0	0
attacked	0	1	0	0	0
border	0	1	0	0	0
early	0	1	0	0	0
from	0	1	0	1	0
gadchiroli	0	1	0	0	0
group	0	1	0	0	0
headquarters	0	1	1	0	0
km	0	1	0	0	0
maharashtra	0	1	0	0	0
monday	0	1	0	0	0
naxalites	0	1	0	0	0
of	0	1	1	0	1
on	0	1	0	0	0
police	0	1	0	0	0
pradesh	0	1	0	0	0

From Text to a Quantitative Matrix

terms	docs				
	1	2	3	4	5
another	1	0	0	0	0
district	1	1	0	0	0
forces	1	0	0	0	0
in	1	0	1	1	1
indian	1	0	0	0	0
kupwara	1	0	0	0	0
militant	1	0	0	0	0
security	1	0	0	0	0
shot	1	0	0	0	0
a	0	4	7	3	5
about	0	1	1	0	0
andhra	0	1	0	0	0
armed	0	1	0	0	0
asarali	0	1	0	0	0
attacked	0	1	0	0	0
border	0	1	0	0	0
early	0	1	0	0	0
from	0	1	0	1	0
gadchiroli	0	1	0	0	0
group	0	1	0	0	0
headquarters	0	1	1	0	0
km	0	1	0	0	0
maharashtra	0	1	0	0	0
monday	0	1	0	0	0
naxalites	0	1	0	0	0
of	0	2	1	0	1
on	0	2	0	0	0
police	0	1	0	0	0
pradesh	0	1	0	0	0

Bernoulli Language Model

- ▶ a document is represented by a feature vector with binary elements taking value 1 if the corresponding word is present in the document and 0 if the word is not present.
- ▶ Let p be the number of words considered, an individual document D_i can be represented as a binary vector $\mathbf{d}_i = (b_{i1}, \dots, b_{ip})$.
- ▶ Then we can represent a collection of n documents as a **term document incidence matrix**.

$$\mathbf{D} = \begin{pmatrix} b_{11} & \dots & b_{1p} \\ b_{21} & \dots & b_{2p} \\ \dots & & \\ b_{n1} & \dots & b_{np} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 1 \\ 0 & 0 & 1 & \dots & 1 \\ 1 & 1 & 0 & \dots & 0 \\ \dots & & & & \end{pmatrix}$$

Bernoulli Language Model

- ▶ Let $P(w_j|Y = k)$ be the probability of word w_j occurring in a document of class k ; the probability of w_j not occurring in a document of this class is given by $(1 - P(w_j|Y = k))$.
- ▶ We can write the document likelihood $P(D_i|k)$ in terms of the individual word likelihoods $P(w_j|Y = k)$:

$$P(D_i|Y = k) = \prod_{j=1}^p P((b_{i1}, \dots, b_{ip})|k)$$

$$\underbrace{\text{NB Assumption}}_{=} \prod_{j=1}^p b_{ij}P(w_j|k) + (1 - b_{ij})(1 - P(w_j|k))$$

- ▶ If word w_j is present, then $b_{ij} = 1$ with probability $P(w_j|Y = k)$
- ▶ We can imagine this as a model for generating document feature vectors of class y , where the document feature vector is modelled as a collection of p weighted coin tosses, the j -th having a probability of success equal to $P(w_j|k)$

Training a Bernoulli Naive Bayes Classifier

- ▶ The parameters of the likelihoods are the probabilities of each word given the document class $P(w_j|Y = k)$; the model is also parameterised by the prior probabilities, $P(Y = k)$.
- ▶ Using a labeled training set of documents we can estimate these parameters. Let n_k be the number of documents of class $Y = k$ in which w_j is observed
- ▶ Let N be the total number of documents, N_k be the total number of documents of class k . Then we can estimate the parameters of the word likelihoods as, for all $j = 1, \dots, p$ and all $k \in \mathcal{C}$.

$$\hat{P}(w_j|k) = \frac{n_k}{N_k}$$

$$\hat{P}(k) = \frac{N_k}{N}$$

Training a Bernoulli Naive Bayes Classifier

Algorithm (*Training a Bernoulli Naive Bayes Classifier*)

1. Define the vocabulary V , where p is the number of words in the vocabulary, i.e. the number of columns.
2. Count the following in the training set:
 - ▶ N the total number of documents
 - ▶ N_k the number of documents labelled with class k , for $k = 1, \dots, |\mathcal{C}|$
 - ▶ $n_k(w_j)$ the number of documents of class $Y = k$ containing word w_j for every class and for each word in the vocabulary.
3. Estimate the likelihoods $P(w_j | Y = k)$ using

$$\hat{P}(w_j | Y = k) = \frac{n_k(w_j)}{N_k}$$

4. Estimate the priors $P(Y = k)$ using

$$\hat{P}(Y = k) = \frac{N_k}{N}$$

Class Assignment using a Bernoulli Naive Bayes Classifier

Once you have trained the Bernoulli Naive Bayes Classifier, you can compute posterior probabilities for a document $D_i = (b_{i1}, \dots, b_{ip})$ using

$$P(Y = k | D_i) = P(Y = k | (b_{i1}, \dots, b_{ip})) \propto \prod_{j=1}^p P((b_{i1}, \dots, b_{ip}) | k) P(k)$$

$$\underbrace{\text{NB Assumption}}_{=}\quad P(k) \left[\prod_{j=1}^p b_{ij} P(w_j | k) + (1 - b_{ij})(1 - P(w_j | k)) \right]$$

for each class $k = 1, \dots, |\mathcal{C}|$ and assign the document to class k that yields the largest posterior probability.

Bernoulli Language Model Use

- ▶ Document model: a document can be thought of as being generated from a multidimensional Bernoulli distribution: the probability of a word being present can be thought of as a (weighted) coin flip with probability $P(w_j|k)$.
- ▶ Document representation: binary vector, elements indicating presence or absence of a word.
- ▶ Multiple occurrences of words: ignored.
- ▶ Behaviour with document length: best for short documents.
- ▶ Behaviour with very common stopwords (such as "the", "a", "here"): since stopwords are present in almost every document, $P(\text{"the"}|k) = 1.0$.
- ▶ Behavior with irrelevant features?

Multi-Nomial Distribution

- ▶ We now introduce another document model, which explicitly takes into account the number of words, so documents are represented as a collection of word counts.
- ▶ The most common distribution to assume is a multinomial distribution, which is a generalization of a Bernoulli distribution.
- ▶ Using all the letters, how many distinct sequences can you make from the word “Mississippi”? There are 11 letters to permute, but “i” and “s” occur four times and “p” twice.
- ▶ If each letter was distinct, you would have 11 choices for first, 10 for second, 9 for third, ... so a total of 11!.
- ▶ However, 4! permutations are identical as the letter “i” is repeated four times; similarly, 4! for “s” and 2! for “p” and 1! for “m”.
- ▶ So the total number of distinct arrangements of the letters that form the word “Mississippi” is:

$$\frac{11!}{4!4!2!1!} = 34650$$

Multi-Nomial Distribution

- ▶ Generally if we have n items of p types (letters or words), with n_1 of type 1, n_2 of type 2 and n_p of type p

$$n_1 + \dots + n_p = n$$

- ▶ then the number of distinct permutations is given by:

$$\frac{n!}{n_1!n_2!\dots n_p!}$$

- ▶ Now suppose a population contains items of $p \geq 2$ different types and that the proportion of items that are of type j is p_j for ($j = 1, \dots, p$), with

$$\sum_{j=1}^p p_j = 1$$

- ▶ Suppose n items are drawn at random (with replacement) and let x_j denote the number of items of type j .
- ▶ The vector $\mathbf{x} = (x_1, \dots, x_p)$ has a multinomial distribution with parameters n and p_1, \dots, p_p .

Multi Nomial Language Model

- ▶ In the multi nomial language model, you model documents as being collections of word counts.
- ▶ Let p be the number of words considered, an individual document D_i can be represented as a vector $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$.
- ▶ $n_i = \sum_{j=1}^p x_{ij}$ is the total number of words of document D_i .
- ▶ Then we can represent a collection of n documents as a **term document count matrix**.

$$\mathbf{X} = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ x_{21} & \dots & x_{2p} \\ \dots & & \\ x_{n1} & \dots & x_{np} \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 & \dots & 1 \\ 0 & 0 & 5 & \dots & 1 \\ 7 & 2 & 1 & \dots & 3 \\ \dots & & & & \end{pmatrix}$$

Multi Nomial Language Model

- ▶ In the Multi Nomial language model, we assume that word counts x_{ij} are generated following a multi nomial distribution.
- ▶ Multi-nomial distribution is a generalization of the Binomial distribution.
- ▶ Let x_i indicate the number of times outcome number i is observed over the n trials, the vector $x_i = (x_{i1}, \dots, x_{ip})$ follows a multinomial distribution with parameters n and p , where $p = (p_1, \dots, p_p)$

$$P(X_1 = x_{i1} \cap \dots \cap X_p = x_{ip}) = \frac{n!}{x_{i1}! \dots x_{ip}!} p_1^{x_{i1}} \dots p_p^{x_{ip}}$$

For example:

$$\mathbf{X} = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ x_{21} & \dots & x_{2p} \\ \dots & & \\ x_{n1} & \dots & x_{np} \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 & \dots & 1 \\ 0 & 0 & 5 & \dots & 1 \\ 7 & 2 & 1 & \dots & 3 \\ \dots & & & & \end{pmatrix}$$

Multi Nomial Language Model

- ▶ We can now write the joint probability of observing a document $D_i = (x_{i1}, \dots, x_{ip})$ of class k as:

$$P(D_i|Y = k) = P((x_{i1}, \dots, x_{ip})|k) = \frac{n_i!}{\prod_{j=1}^p x_{ij}!} \prod_{j=1}^p P(w_j|k)^{x_{ij}}$$

$$\propto \prod_{j=1}^p P(w_j|k)^{x_{ij}}$$

- ▶ $n_i = \sum_{j=1}^p x_{ij}$ is the total number of words of document i .
- ▶ So we need to estimate $P(w_j|k)$ from our training data.
- ▶ We generally ignore the scaling factor $\frac{n_i!}{\prod_{j=1}^p x_{ij}!}$, because it is not a function of the class k ! So we can safely ignore it, as it will not affect the optimal class assignment.

Multinomial Model and Estimation of Likelihoods

- ▶ For the Multinomial distribution we need to estimate the vector of $P(w_j|k)$ for all $j = 1, \dots, p$ and all $k \in \mathcal{C}$.
- ▶ I.e. there is a different vector $\mathbf{p} = (p_1, \dots, p_p)$ for every possible class k .
- ▶ The maximum likelihood estimator turns out to be slightly more tricky:

$$P(w_j|k) = \frac{\text{No. of times word } w_j \text{ appears in all documents of class } k}{\text{Total No. of words in all documents of class } k}$$

- ▶ A more formal notation is

$$\hat{P}(w_j|Y = k) = \frac{\sum_{i=1}^N x_{ij} z_{ik}}{\sum_{s=1}^p \sum_{i=1}^N x_{is} z_{ik}}$$

where z_{ik} is a dummy variable that is 1, in case document i has class k .

Training a Multinomial Naive Bayes Classifier

Algorithm (*Training a Multinomial Naive Bayes Classifier*)

1. Define the vocabulary V , where p is the number of words in the vocabulary, i.e. the number of columns.
2. Count the following in the training set:
 - ▶ N the total number of documents
 - ▶ N_k the number of documents labelled with class k , for $k = 1, \dots, |C|$
 - ▶ x_{ij} the frequency of word w_j in document D_i , computed for every word w_j in V .
3. Estimate the likelihoods $P(w_j|Y = k)$ using

$$\hat{P}(w_j|Y = k) = \frac{\sum_{i=1}^N x_{ij} z_{ik}}{\sum_{s=1}^p \sum_{i=1}^N x_{is} z_{ik}}$$

4. Estimate the priors $P(Y = k)$ using

$$\hat{P}(Y = k) = \frac{N_k}{N}$$

Class Assignment using a Multinomial Naive Bayes Classifier

We compute the posterior probability of a document D_i , represented as a word count vector \mathbf{x}_i , belonging to some class k as:

$$P(Y = k | D_i) = P(Y = k | \mathbf{x}_i) \underbrace{\propto}_{\text{Bayes Law}} P((x_{i1}, \dots, x_{ip}) | k) P(k) \\ \underbrace{\propto}_{\text{NB Assumption}} P(k) \prod_{j=1}^p P(w_j | k)^{x_{ij}}$$

Note that, unlike the Bernoulli model, words that do not occur in a document (i.e., for which $x_{ij} = 0$) do not affect the probability (since $p^0 = 1$).

Thus we can write the posterior probability in terms of the set of words U that appear in document i , i.e. the set of words U defined by $x_{ij} > 0$.

A Worked Example

Suppose you want to build a sentiment classifier for feedbacks received on a website. You have the following training data.

- [+] "very good site"
- [+] "nothing very happy so far"
- [+] "nothing site is excellent"
- [-] "free postage throughout"
- [-] "lower the price"
- [-] "lower price not enough gift deals"
- [-] "dont bring this box up"
- [-] "dont have a pop up box"

So we observe 8 feedbacks, 3 are positive while 5 are negative. Lets represent these documents as a term document matrix.

A Worked Example

TDM

```
##          docs
## terms    1 2 3 4 5 6 7 8
##  a       0 0 0 0 0 0 0 1
##  box      0 0 0 0 0 0 1 1
##  bring     0 0 0 0 0 0 1 0
##  deals     0 0 0 0 0 1 0 0
##  dont      0 0 0 0 0 0 1 1
##  enough    0 0 0 0 0 1 0 0
##  excellent 0 0 1 0 0 0 0 0
##  far       0 1 0 0 0 0 0 0
##  free      0 0 0 1 0 0 0 0
##  gift      0 0 0 0 0 1 0 0
##  good      1 0 0 0 0 0 0 0
##  happy     0 1 0 0 0 0 0 0
##  have      0 0 0 0 0 0 0 1
##  is        0 0 1 0 0 0 0 0
##  lower     0 0 0 0 1 1 0 0
##  not       0 0 0 0 0 1 0 0
##  nothing   0 1 1 0 0 0 0 0
##  pop       0 0 0 0 0 0 0 1
##  postage   0 0 0 1 0 0 0 0
##  price     0 0 0 0 1 1 0 0
##  site      1 0 1 0 0 0 0 0
##  so        0 1 0 0 0 0 0 0
##  the       0 0 0 0 1 0 0 0
##  this      0 0 0 0 0 0 1 0
##  throughout 0 0 0 1 0 0 0 0
##  up        0 0 0 0 0 0 1 1
##  very      1 1 0 0 0 0 0 0
```

A Worked Example

Given the training data, we can easily get the priors:

$$\hat{P}(+) = 3/8$$

$$\hat{P}(-) = 5/8$$

In total, we have 27 different words. We now need to compute a total of 27×2 class conditional probabilities $P(w_j|k)$.

$$P(w_j|k) = \frac{\text{No. of times word } w_j \text{ appears in all documents of class } k}{\text{Total No. of words in all documents of class } k}$$

In order to compute these class conditional likelihoods, we need a tabulation of words.

A Worked Example

WF

##	terms	Freq	CountNegative	CountPositive
## 1:	box	2	2	0
## 2:	dont	2	2	0
## 3:	lower	2	2	0
## 4:	nothing	2	0	2
## 5:	price	2	2	0
## 6:	site	2	0	2
## 7:	up	2	2	0
## 8:	very	2	0	2
## 9:	a	1	1	0
## 10:	bring	1	1	0
## 11:	deals	1	1	0
## 12:	enough	1	1	0
## 13:	excellent	1	0	1
## 14:	far	1	0	1
## 15:	free	1	1	0
## 16:	gift	1	1	0
## 17:	good	1	0	1
## 18:	happy	1	0	1
## 19:	have	1	1	0
## 20:	is	1	0	1
## 21:	not	1	1	0
## 22:	pop	1	1	0
## 23:	postage	1	1	0
## 24:	so	1	0	1
## 25:	the	1	1	0
## 26:	this	1	1	0
## 27:	throughout	1	1	0
##	terms	Freq	CountNegative	CountPositive

Laplace Smoothing

When we estimate the class conditional likelihoods, we may run into the problem that for some class k and some word feature w_j , $\hat{P}(w_j|k) = 0$. This is problematic, since we compute:

$$P(Y = k|D_i) \propto P(k) \prod_{j=1}^p P(w_j|k)^{x_{ij}}$$

In case in the training data, some estimate $\hat{P}(w_j|k) = 0$, but a word appears in the prediction document D_i , you would end up with a probability of $P(Y = k|D_i) = 0$.

Our estimate

$$\hat{P}(w_j|Y = k) = \frac{\sum_{i=1}^N x_{ij} z_{ik}}{\sum_{s=1}^p \sum_{i=1}^N x_{is} z_{ik}}$$

underestimates the likelihoods of words that do not occur in the training data. Even if word w is not observed for class $Y = k$ in the training set, we would still like $P(w|Y = k) > 0$.

Laplace Smoothing

Since all the $\sum_{j=1}^p P(w_j|k) = 1$, if unobserved words have underestimated probabilities, then those words that are observed must have overestimated probabilities. Therefore, one way to alleviate the problem is to remove a small amount of probability allocated to observed events and distribute this across the unobserved events. A simple way to do this, sometimes called Laplace's law of succession or add one smoothing, adds a count of one to each word type.

$$\hat{P}(w_j|Y = k) = \frac{1 + \sum_{i=1}^N x_{ij} z_{ik}}{p + \sum_{s=1}^p \sum_{i=1}^N x_{is} z_{ik}}$$

A word \tilde{j} that does not appear in some class k now gets an estimate of

$$\hat{P}(w_{\tilde{j}}|k) = \frac{1}{p + \sum_{s=1}^p \sum_{i=1}^N x_{is} z_{ik}}$$

A Worked Example

```
WF$CountNegative<-WF$CountNegative+1
```

```
WF$CountPositive<-WF$CountPositive+1
```

```
WF$wjneg <- WF$CountNegative/sum(WF$CountNegative)
```

```
WF$wjpos <- WF$CountPositive/sum(WF$CountPositive)
```

```
WF
```

##	terms	Freq	CountNegative	CountPositive	wjneg	wjpos
## 1:	box	2	3	1	0.06	0.0256
## 2:	dont	2	3	1	0.06	0.0256
## 3:	lower	2	3	1	0.06	0.0256
## 4:	nothing	2	1	3	0.02	0.0769
## 5:	price	2	3	1	0.06	0.0256
## 6:	site	2	1	3	0.02	0.0769
## 7:	up	2	3	1	0.06	0.0256
## 8:	very	2	1	3	0.02	0.0769
## 9:	a	1	2	1	0.04	0.0256
## 10:	bring	1	2	1	0.04	0.0256
## 11:	deals	1	2	1	0.04	0.0256
## 12:	enough	1	2	1	0.04	0.0256
## 13:	excellent	1	1	2	0.02	0.0513
## 14:	far	1	1	2	0.02	0.0513
## 15:	free	1	2	1	0.04	0.0256
## 16:	gift	1	2	1	0.04	0.0256
## 17:	good	1	1	2	0.02	0.0513
## 18:	happy	1	1	2	0.02	0.0513
## 19:	have	1	2	1	0.04	0.0256
## 20:	is	1	1	2	0.02	0.0513
## 21:	not	1	2	1	0.04	0.0256
## 22:	pop	1	2	1	0.04	0.0256
## 23:	postage	1	2	1	0.04	0.0256
## 24:	so	1	1	2	0.02	0.0513
## 25:	the	1	2	1	0.04	0.0256
## 26:	this	1	2	1	0.04	0.0256
## 27:	throughout	1	2	1	0.04	0.0256
##	terms	Freq	CountNegative	CountPositive	wjneg	wjpos

Scoring a document

Compute

$$P(Y = k|D_i) \propto P(k) \prod_{j=1}^p P(w_j|k)^{x_{ij}}$$

for document "you dont have good deals".

```
doc<-"you dont have good deals"
docwords<-wordfreq(doc)
setnames(docwords, "Freq", "xij")

WF.score<-join(WF,docwords)[!is.na(xij)]

## Joining by: terms
WF.score
##      terms Freq CountNegative CountPositive wjneg  wjpos docs xij
## 1: dont    2          3           1 0.06 0.0256    1  1
## 2: deals    1          2           1 0.04 0.0256    1  1
## 3: good     1          1           2 0.02 0.0513    1  1
## 4: have     1          2           1 0.04 0.0256    1  1

WF.score[, list(neg=wjneg^xij, pos= wjpos^xij)][, list(neg=prod(neg), pos=prod(pos))] * c(5/8,3/8)

##      neg      pos
## 1: 1.2e-06 7.2e-07
```

Typically transform to logs

As with other likelihood methods, we generally transform everything into log space.

I.e. we compute posterior probabilities as a weighted sum of the logs of individual word scores:

$$\log(P(Y = k|D_i)) \propto P(k) + \sum_{j=1}^p x_{ij} \log P(w_j|k)$$

Pre defined vocabularies

- ▶ In some situations, building an own vocabulary to be used for classification tasks based on text data may be impossible due to a lack of training data or simply unnecessary.
- ▶ Often times its possible to work with *sentiment lexicons*, lists of words that are pre- annotated with positive or negative sentiment.
- ▶ Four popular lexicons are the General Inquirer (Stone et al., 1966), LIWC (Pennebaker et al., 2007), the opinion lexicon LIWC of Hu and Liu (2004) and the MPQA Subjectivity Lexicon (Wilson et al., 2005).
- ▶ For example the MPQA subjectivity lexicon has 6885 words, 2718 positive and 4912 negative, each marked for whether are strongly or weakly biased.

General Implications of the “Naive” Bayes Assumption

- ▶ Word order does not matter: whether the word “love” appears at the beginning of a sentence or at the end is ignored.
- ▶ Words are independently drawn, so we ignore multi word fragments such as negations “not good”.
- ▶ Multinomial does not ignore fact that words appearing multiple times may be indicative.
- ▶ Typically, one would prefer Bernoulli language model for short documents, whereas Multinomial model is adequate for longer texts.
- ▶ Nevertheless, despite this stark assumption, Naive Bayes performs very well in a lot of cases.

Plan

Text as Data

Naive Bayes illustration, validation, learning curve

Using Machines to (help) Code Conflict Data

- ▶ In my paper Social Insurance and Conflict: Evidence from India, I rely on several machine learning methods to automatically construct a conflict data set.
- ▶ **Problem:** Human coding is subject to subjectivity bias, not every person would code some piece of text in the same way. This is a problem due to reduced *transparency*.
- ▶ **Solution:** Supervised Machine learning algorithms only require a single training set to be coded by humans, the training data - at some level - is the only input.
- ▶ My paper constructs a conflict dataset, using various **Natural Language Processing** methods, including Naive Bayes classifiers.
- ▶ The raw data consists of 50,000 newspaper clippings about events in South Asia, which needed to be converted into a dataset providing the number of conflict events per district.

Leveraging the cloud to build a training data set



The screenshot shows the CrowdFlower website. At the top is a dark navigation bar with the CrowdFlower logo and links for Products, Company, Blog, Docs, and a Log in button. Below the navigation bar is the main heading "The World's Largest Workforce" in large orange text, followed by the tagline "Instantly hire millions of people to collect, filter, and enhance your data." in grey. The page features three dark grey boxes on the left: "RTFM Real Time Foto Moderator" for crowdsourced image moderation, and "Senti Sentiment Analysis" for fast, accurate human review of social media content. On the right is a large graphic of a world map with a grid. Overlaid on the map are two data boxes: "Real-time Crowd Labor" with a person icon, and a box showing "— judgments/sec current velocity" and "515,107,994 total judgments".

CrowdFlower Products Company Blog Docs Log in

The World's Largest Workforce

Instantly hire millions of people to collect, filter, and enhance your data.

RTFM
Real Time Foto Moderator
Crowdsourced image moderation with a simple real-time API.

Senti
Sentiment Analysis
Fast, accurate human review of user-generated social media content.


Real-time Crowd Labor

— judgments/sec
current velocity

515,107,994
total judgments

<http://www.crowdflower.com>

Leveraging the cloud to build a training data set


 Thiemo Fetzner ▾

Create a new job


Select a template or start from scratch

What would you like to do?


RECENT
Rate the Relevance of a Search Result
[Use this template](#)




Sentiment Analysis




Search Relevance



Data Categorization



Data Collection & Enrichment



Data Validation






Image Annotation




Transcription



Content Moderation

<http://www.crowdfunder.com>



Leveraging the cloud to build a training data set

The screenshot displays the CrowdFlower Design interface for creating a task. The top navigation bar includes tabs for DATA, DESIGN (active), QUALITY, LAUNCH, MONITOR, and RESULTS. The user's name, Thileno Fetzner, is in the top right.

Design [Add -] STATUS: **Finished** [View Results]

Title

Tag Short Sentences

Content

DATA | {{sentence}}

National Liberation Front of Tripura (NLFT) militants killed four persons and injured four others in Ramdulal Para, North Tripura district.

DATA

DATA | {{act}}

Who was injured ?

QUESTION | multiple choice

Choose one

- ☐ Terrorist
- ☐ Civilian
- ☐ Security

Mark the best matching case.

RIGHT SIDEBAR:

JOB ID 195954

Tag Short Sentences

⚙ Settings

🔍 Preview

📄 Copy -

READ THE DOCS

- [Code Editor Guide](#)
- [Code Editor Tips](#)
- [CrowdFlower Markup Language](#)
- [CML Attributes Guide](#)

<http://www.crowdfLOWER.com>

Leveraging the cloud to build a training data set

Two unidentified terrorists massacred six members of a family and left a seventh injured at Mangnar Top, Poonch district, on December 31, 2001.

Who was Injured?

Choose one

- ☐ Terrorist
- ☐ Civilian
- ☐ Security

 Mark the best matching case.

`http://www.crowdfunder.com`

What do we want to classify?

```
TTIM[100:120]$objectclean
```

```
## [1] "women"
## [2] "security forces"
## [3] "former terrorist Hizb - ul"
## [4] "NLFT terrorist"
## [5] "tribal headmaster"
## [6] "civilian"
## [7] "Special Police Officer ( SPO )"
## [8] "civilian"
## [9] "ULFA deputy chief"
## [10] "political leader ruling National Conference"
## [11] "suspected police informer"
## [12] ", founding member and perpetrators hijacking Indian Airlines plane to"
## [13] "people attack army base city 's"
## [14] "terrorist"
## [15] "civilian , reported to be former local level politician"
## [16] "Border Security Force personnel"
## [17] "terrorists Hizb - ul Mujhadeen ( HuM ) and security force personnel"
## [18] "suspected ULFA terrorist"
## [19] "terrorist Lashkar - e ( LeT )"
## [20] ", deputy kilonser ( minister ) outfit corporals arms ammunition and large amount extortion cash"
## [21] "terrorist belonging to Jaish - e residence political leader national party February"
```

Naive Bayes Classifier with RTextTools and e1071 package

```
##just look at one word feature
L1 <- create_matrix(c(TTIM[,paste(objectcleanpp,sep=" ")]),
                    language="english",stemWords=FALSE)
##CREATION OF NON SPARSE MATRIX
DTM<-as.matrix(L1)

dim(DTM)
## [1] 1397 2460

##changing column names
colnames(DTM) <- paste("stem_", colnames(DTM),sep="")

##turn this into a document-term-incidence matrix
DTM<-apply(DTM, 2, function(x) as.factor(x>0))

TTIM2<-cbind(TTIM,DTM)
TTIM2$label1<-as.factor(TTIM2$label1)

TRAINING<-TTIM2[!is.na(label1)]
set.seed(2016)

VALIDATION<-sample(1:nrow(TRAINING), 200)
```

Obtaining a Naive Bayes Classifier

```
##just look at one word feature

model <- naiveBayes(label1 ~ stem_terrorist , data = TRAINING[-VALIDATION])

model

##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   civilian  security terrorist
##      0.263    0.167    0.570
##
## Conditional probabilities:
##           stem_terrorist
## Y           FALSE  TRUE
## civilian  1.000 0.000
## security  0.970 0.030
## terrorist 0.894 0.106
```


Interpreting Output

Naive Bayes Classifier for Discrete Predictors

Call:

```
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

A-priori probabilities:

```
Y
  civilian  security  terrorist
0.2631579 0.1670844 0.5697577
```

Conditional probabilities:

	stem_terrorist	
Y	FALSE	TRUE
civilian	1.0000000	0.0000000
security	0.9450000	0.0550000
terrorist	0.7844575	0.2155425

Interpreting Output

Naive Bayes Classifier for Discrete Predictors

Call:

```
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

A-priori probabilities:

Y	
	civilian security terrorist
	0.2631579 0.1670844 0.5697577

$$\hat{P}(Y = k)$$

Conditional probabilities:

		stem_terrorist	
Y		FALSE	TRUE
	civilian	1.0000000	0.0000000
	security	0.9450000	0.0550000
	terrorist	0.7844575	0.2155425

Interpreting Output

Naive Bayes Classifier for Discrete Predictors

Call:

```
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

A-priori probabilities:

```
Y
  civilian  security terrorist
0.2631579 0.1670844 0.5697577
```

Conditional probabilities:

	stem_terrorist	
Y	FALSE	TRUE
civilian	1.0000000	0.0000000
security	0.9450000	0.0550000
terrorist	0.7844575	0.2155425

$$\hat{P}(\text{word terrorist}|k)$$

Naive Bayes Classifier with quanteda package

```
##just look at one word feature
library(quanteda)
TTIM<-TTIM[order(sid)]
set.seed(06022017)
##make sure TTIM has (same) random order
TTIM<-TTIM[sample(1:nrow(TTIM), nrow(TTIM))]
TTIM[, label1 := factor(label1)]
L1 <- corpus(c(TTIM[,paste(objectcleanpp,sep=" ")]))
L1.maindfm <- dfm(L1)

##same dimensionality
dim(L1.maindfm)

## [1] 1397 2546

trainingclass <- TTIM$label1
#first 200 obs for validation set
trainingclass[1:200]<-NA
NB<- textmodel(L1.maindfm, trainingclass, model = "NB")

##CREATION OF NON SPARSE MATRIX
pred<-predict(NB, newdata = L1.maindfm[1:200, ])

#confusion table
table(pred$nb.predicted,TTIM[1:200]$label1 )

##
##          civilian security terrorist
## civilian      44         2         8
## security      4         36         6
## terrorist      6         0        94

sum(diag(3) * table(pred$nb.predicted,TTIM[1:200]$label1 ))/200

## [1] 0.87

##accuracy around 87%
```

Naive Bayes Classifier with quanteda package

```
##class conditional posterior probabilities
t(NB$PcGw)[1:15,]

## 15 x 3 Matrix of class "dgeMatrix"
##               docs
## features      terrorist civilian security
##   slain      0.199    0.373    0.4278
##   activist    0.122    0.825    0.0526
##   brother     0.205    0.576    0.2199
##   and         0.348    0.291    0.3607
##   relative    0.199    0.373    0.4278
##   landmine    0.199    0.373    0.4278
##   blasts      0.205    0.576    0.2199
##   rifleman    0.199    0.373    0.4278
##   identified  0.692    0.240    0.0687
##   mohmmud     0.199    0.373    0.4278
##   slatur      0.199    0.373    0.4278
##   rahman      0.114    0.641    0.2450
##   lady        0.199    0.373    0.4278
##   accomplice  0.253    0.475    0.2721
##   ningombam   0.498    0.234    0.2679

NB$Pc

## terrorist  civilian  security
##    0.333    0.333    0.333

##current quanteda implementation has a bug
NB<- textmodel_NB(L1.maindfm, trainingclass,prior = "docfreq")
NB$Pc

## terrorist  civilian  security
##    0.566    0.267    0.166

prop.table(table(trainingclass))

## trainingclass
##   civilian  security terrorist
##    0.267    0.166    0.566
```

Bernoulli vs Multinomial

```
NB<- textmodel_NB(L1.maindfm, trainingclass,distribution="Bernoulli",smooth=1)
pred<-predict(NB, newdata = L1.maindfm[1:200, ])
#confusion table
table(pred$nb.predicted,TTIM[1:200]$label1 )

##
##          civilian security terrorist
## civilian      44         2         8
## security       4        36         6
## terrorist      6         0        94

sum(diag(3) * table(pred$nb.predicted,TTIM[1:200]$label1 ))/200
## [1] 0.87

NB<- textmodel_NB(L1.maindfm, trainingclass,distribution="multinomial",smooth=1)
pred<-predict(NB, newdata = L1.maindfm[1:200, ])
#confusion table
table(pred$nb.predicted,TTIM[1:200]$label1 )

##
##          civilian security terrorist
## civilian      44         2         8
## security       4        36         6
## terrorist      6         0        94

sum(diag(3) * table(pred$nb.predicted,TTIM[1:200]$label1 ))/200
## [1] 0.87
```

Bias vs Variance Trade-Off

There are two aspects of the performance of a classifier trained on a finite sample size n :

1. **bias**, i.e. on average a classifier trained on a finite training sample is worse than the classifier trained with larger number of training cases
2. **variance** different training sets may give quite different model performance. Even with few cases, you may be lucky and get good results. Or you have bad luck and get a really bad classifier.

Sensitivity of model fit to specific training vs validation data set used

```
##just look at one word feature
TTIM<-TTIM[order(sid)]
set.seed(06022017)

ACC<-NULL
for(i in 1:50) {
  ##randomly reorder
  TTIM<-TTIM[sample(1:nrow(TTIM), nrow(TTIM))]

  L1 <- corpus(c(TTIM[,paste(objectcleanpp,sep=" ")]))
  L1.dfm <- dfm(L1)

  trainingclass <- factor(TTIM$label1, ordered = TRUE)

  #first 200 obs for validation set
  trainingclass[1:200]<-NA

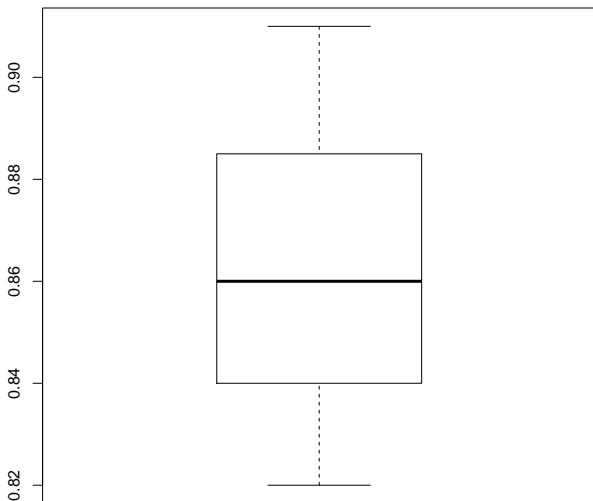
  NB<- textmodel(L1.dfm, trainingclass, model = "NB")
  ##CREATION OF NON SPARSE MATRIX

  pred<-predict(NB, newdata = L1.dfm[1:200, ])

  #confusion table
  ACC<-c(ACC,sum(diag(3) * table(pred$nb.predicted,TTIM[1:200]$label1 ))/200)
}

boxplot(ACC)
```


Sensitivity of model fit to specific training vs validation data set used



Building a training data set

- ▶ Feature vocabularies often exist: profanity, sentiment lexicons, etc.
- ▶ Biggest impediment is often the lack of a training data set.
- ▶ For most our applications, it may involve you actually hand coding some observations.
- ▶ But how big should the training data set be?

Larger Training data set? Learning curves

A model's "learning curve", gives the (average) model performance as function of the training sample size. As you can guess, learning curves depend on a lot of things, e.g.

- ▶ classification method
- ▶ complexity of the classifier
- ▶ how well the classes are separated.

How do things look in this particular context? We can evaluate the learning curve by simply plotting out the performance of the model as we expand the size of the training set.

"Estimating" the Learning Curve

```
##just look at one word feature
TTIM<-TTIM[order(sid)]
set.seed(06022017)
##make sure TTIM has random order
TTIM<-TTIM[sample(1:nrow(TTIM), nrow(TTIM))]

LEARNING<-NULL
for(i in seq(25,1200, 25)) {
  ##randomly reorder

  TEMP<-TTIM[c(1:200,201:(200+i))]

  L1 <- corpus(TEMP[,paste(objectcleanpp,sep=" ")])
  L1.dfm <- dfm(L1)

  trainingclass <- factor(TEMP$label1, ordered = TRUE)

  #first 200 obs for validation set
  trainingclass[1:200]<-NA

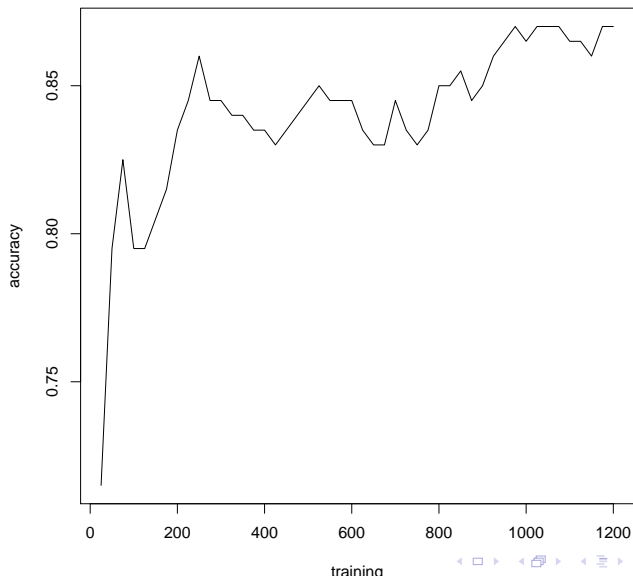
  NB<- textmodel(L1.dfm, trainingclass, model = "NB")
  ##CREATION OF NON SPARSE MATRIX

  pred<-predict(NB, newdata = L1.dfm[1:200, ])

  #confusion table
  LEARNING<-rbind(LEARNING,data.frame(training=i,
                                         accuracy=sum(diag(3) * table(pred$nb.predicted,TTIM[1:200]$label1 ))/200),
  }

  plot(LEARNING, type="l")
```

"Estimating" the Learning Curve



Comparing Performance between Logistic Regression and Naive Bayes

- ▶ In most cases, we obtain multiple different classifiers and study how the performance of either of them differs.
- ▶ Last week, we have applied Maximum Entropy (Logistic regression) to this data.
- ▶ Logistic regression may be more vulnerable to the inclusion of irrelevant features (higher variance)
- ▶ Do we see this play out in the actual data?

MaxEnt and NaiveBayes produce similar results

```
library(tm)
library(RTextTools)
TTIM<-TTIM[order(sid)]
set.seed(06022017)
##make sure TTIM has (same) random order
TTIM<-TTIM[sample(1:nrow(TTIM), nrow(TTIM))]
TTIM$validation<- 0
TTIM[1:200]$validation<-1
##we are not removing stopwords or doing anything to be comparable to NaiveBayes
DOC<-create_matrix(TTIM[,paste(objectcleanpp,sep=" ")] ,language="english")
dim(DOC)

## [1] 1397 2460

##create container does not like factors
DOCCONT<-create_container(DOC,as.numeric(as.factor(TTIM$label1)), trainSize=201:1200,
                          testSize=1:200, virgin=TRUE)
MOD <- train_models(DOCCONT, algorithms=c("MAXENT"))
RES <- classify_models(DOCCONT, MOD)
analytics <- create_analytics(DOCCONT, RES)
res<-data.table(analytics@document_summary)
VALID<-cbind(TTIM[validation==1],res)
#confusion matrix
table(VALID$label1,factor(VALID$MAXENTROPY_LABEL, labels=levels(VALID$label1)))

##
##          civilian security terrorist
## civilian      44         1         9
## security       4        34         0
## terrorist       9         2        97

sum(diag(3) *table(VALID$MAXENTROPY_LABEL,VALID$label1))/200

## [1] 0.875
```

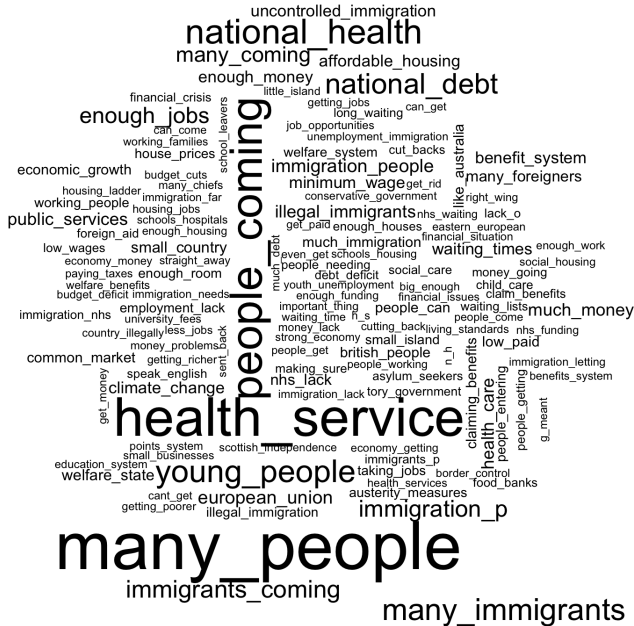
Going back to the Brexit example

- ▶ The 2015 British Election study asked "what is the single most important issue that the UK is facing at the moment"?
- ▶ Free text response of thousands of respondents
- ▶ Can we use the responses given to capture meaningful variation in the stated voting intention in the question "how would you vote in a referendum on UK's EU membership"?

Single biggest issue facing the UK... unigram model



Single biggest issue facing the UK... bigram model



Going back to the Brexit example

```
##just look at one word feature
table(DTA[,list(toomanyimmigrants, leaveeu)])

##               leaveeu
## toomanyimmigrants leave remain
##               0      60    478
##               1     872    777
```

Did not achieve good results using logistic regression. How do we do with Naive Bayes?

Going back to the Brexit example

```
##just look at one word feature
docnames(C)<-DTA$finalserialno
C.dfm<-dfm(C, tolower = TRUE, stem = TRUE, removeNumbers=TRUE,removePunct=TRUE, remove = stopwords("english"))

validation <- factor(DTA$leaveeu, ordered = TRUE)

#first 200 obs for validation set
validation[1:200]<-NA

NB<- textmodel(C.dfm, validation, model = "NB",distribution="Bernoulli")
##CREATION OF NON SPARSE MATRIX

pred<-predict(NB, newdata = C.dfm[1:200, ])

##much better...
table("prediction"=pred$nb.predicted,"data"=DTA[1:200]$leaveeu)

##           data
## prediction leave remain
##      leave      59      40
##      remain     31      70
```

Going back to the Brexit example

```
##just look at one word feature
docnames(C)<-DTA$finalserialno
C.dfm<-dfm(C, tolower = TRUE, stem = TRUE, removeNumbers=TRUE,removePunct=TRUE, remove = stopwords("english"))

validation <- factor(DTA$leaveeu, ordered = TRUE)

#first 200 obs for validation set
validation[1:200]<-NA

NB<- textmodel(C.dfm, validation, model = "NB",distribution="Bernoulli")
##CREATION OF NON SPARSE MATRIX

pred<-predict(NB, newdata = C.dfm[1:200, ])

##much better...
table("prediction"=pred$nb.predicted,"data"=DTA[1:200]$leaveeu)

##           data
## prediction leave remain
##      leave      59      40
##      remain     31      70
```

Looking at class posteriors $P(C|w)$

```
##just look at one word feature
```

```
t(NB$PcGw)[1:20,]
```

```
## 20 x 2 Matrix of class "dgeMatrix"
```

```
##          docs
```

```
## features  leave remain
```

```
##   stabilis 0.566 0.434
```

```
##   economi 0.321 0.679
```

```
##   nhs      0.455 0.545
```

```
##   immigr  0.691 0.309
```

```
##   popul   0.544 0.456
```

```
##   control 0.514 0.486
```

```
##   refuge   0.511 0.489
```

```
##   go       0.434 0.566
```

```
##   somewher 0.566 0.434
```

```
##   illeg    0.624 0.376
```

```
##   need     0.474 0.526
```

```
##   job      0.395 0.605
```

```
##   lack     0.409 0.591
```

```
##   general  0.525 0.475
```

```
##   islam    0.662 0.338
```

```
##   extrem   0.566 0.434
```

```
##   kill     0.395 0.605
```

```
##   peopl    0.491 0.509
```

```
##   take     0.587 0.413
```

```
##   innoc    0.566 0.434
```

Looking at class posteriors $P(C|w)$

```
##just look at one word feature
```

```
t(NB$PcGw)[1:20,]
```

```
## 20 x 2 Matrix of class "dgeMatrix"
```

```
##          docs
```

```
## features  leave remain
```

```
##   stabilis 0.566 0.434
```

```
##   economi 0.321 0.679
```

```
##   nhs      0.455 0.545
```

```
##   immigr  0.691 0.309
```

```
##   popul   0.544 0.456
```

```
##   control 0.514 0.486
```

```
##   refuge   0.511 0.489
```

```
##   go       0.434 0.566
```

```
##   somewher 0.566 0.434
```

```
##   illeg    0.624 0.376
```

```
##   need     0.474 0.526
```

```
##   job      0.395 0.605
```

```
##   lack     0.409 0.591
```

```
##   general  0.525 0.475
```

```
##   islam    0.662 0.338
```

```
##   extrem   0.566 0.434
```

```
##   kill     0.395 0.605
```

```
##   peopl    0.491 0.509
```

```
##   take     0.587 0.413
```

```
##   innoc    0.566 0.434
```

Looking at class posteriors $P(C|w)$

##just look at one word feature

pred

Predicted textmodel of type: Naive Bayes

(showing 30 of documents)

##

##	lp(leave)	lp(remain)	Pr(leave)	Pr(remain)	Predicted
## 47208	-14.1739	-13.6892	0.3811	0.6189	remain
## 60906	-5.0304	-4.8511	0.4553	0.5447	remain
## 32707	-3.3715	-4.1752	0.6908	0.3092	leave
## 34906	-53.9277	-55.4627	0.8227	0.1773	leave
## 59104	-21.7903	-20.6644	0.2449	0.7551	remain
## 11802	-3.3715	-4.1752	0.6908	0.3092	leave
## 19905	-47.9658	-48.7015	0.6761	0.3239	leave
## 30701	-6.0520	-5.8521	0.4502	0.5498	remain
## 44503	-173.1907	-171.6101	0.1707	0.8293	remain
## 31703	-34.6634	-34.9078	0.5608	0.4392	leave
## 63609	-68.5721	-66.4294	0.1050	0.8950	remain
## 47504	-3.3715	-4.1752	0.6908	0.3092	leave
## 51306	-113.7691	-117.3244	0.9722	0.0278	leave
## 63306	-6.7829	-8.0038	0.7722	0.2278	leave
## 62506	-14.7539	-15.8054	0.7411	0.2589	leave
## 29908	-6.0520	-5.8521	0.4502	0.5498	remain
## 24408	-20.5260	-18.3752	0.1043	0.8957	remain
## 38205	-0.6931	-0.6931	0.5000	0.5000	leave
## 43206	-15.3635	-16.1355	0.6840	0.3160	leave
## 27601	-38.0005	-41.1666	0.9595	0.0405	leave
## 63606	-72.3237	-76.8300	0.9891	0.0109	leave
## 29401	-3.3715	-4.1752	0.6908	0.3092	leave
## 45508	-44.9853	-45.2988	0.5777	0.4223	leave
## 42209	-25.9966	-27.0479	0.7410	0.2590	leave
## 65501	-13.1197	-11.8588	0.2208	0.7792	remain
## 12702	-91.3996	-88.3762	0.0464	0.9536	remain
## 64610	-3.3715	-4.1752	0.6908	0.3092	leave
## 45803	-53.7236	-56.8970	0.9598	0.0402	leave
## 10808	-32.7714	-35.5885	0.9436	0.0564	leave
## 37309	-66.8873	-68.3987	0.8193	0.1807	leave