

Topic Modelling in Irish Blogs

Final Year Project Report

Sam Power

Project ID: 27813

A final report submitted in part fulfilment of the degree of BSc (Hons.)
in the School of Computer Science and Informatics with the supervision
of Prof. Padraig Cunningham and moderated by Derek Greene.



School of Computer Science and Informatics

University College Dublin

12 March 2014

Project Specification

General Information:

The idea of Latent Semantic Analysis has been around for a number of years. It entails mapping texts onto latent topics so that texts that are not superficially similar will be recognised to be related if they map to some of the same latent topics. Topic modelling has become popular in recent years in social media analysis as a mechanism for modelling the huge volume of user generated content that is available. This popularity of topic modelling has been supported by the availability of tools such as Mallet and Gensim.

Insight Centre researchers at UCD have a collection of 42,000 web pages from ~800 Irish blogs which were nominated for Blog Awards Ireland in 2013. The blogs are annotated with category labels (e.g. Music, Photoblogs, Diaspora, Fashion etc). The objective of this project would be to do some topic modelling of this collection.

Mandatory:

- Parse HTML from blog pages (e.g. using BeautifulSoup in Python)
- Separate out the blog post content from the template/non-content.
- Build a bag-of-words model (e.g. using Gensim).
- Apply LDA topic modelling (e.g. using Gensim or Mallet). Do the topics correspond to the awards categories? Or are we seeing something different?

Discretionary:

- Apply NMF for topic modelling. Compare the results from LDA & NMF, and also compare to the awards categories.
- Apply entity extraction to blog text. Can the addition of entities as features (e.g. Irish place names, celebrities, band names etc) improve topic modelling?

Exceptional:

- Come up with a rule-of-thumb for selecting the number of topics for LDA and/or NMF?
- Look at the difference between LDA & NMF topic modelling in more detail, if one is more effective than the other - why is that?

Table of Contents

Abstract.....	4
1 Introduction	5
2 Background Research	6
2.1 Text Extraction	6
2.1.1 Boilerpipe API.....	7
2.2 Topic Models.....	8
2.2.1 Latent Dirichlet Allocation (LDA).....	8
2.2.2 Non-Negative Matrix Factorization (NMF)	10
3 Implementation	12
3.1 Boilerpipe API	12
3.2 Generating the Stoplists.....	13
3.3 Choosing the Number of Topics.....	15
3.4 LDA - MALLET	18
3.5 NMF - scikit-learn	19
3.6 Scoring the Topics	19
4 Results	20
4.1 Conclusion	21
4.2 Future Work	21
5 Acknowledgments.....	22
6 Bibliography	23

Abstract

In this paper we focus on the concept of using machine learning techniques to accurately predict the categories of the Blog Awards Ireland 2013 blogs. We explore the use of Latent Dirichlet Allocation (LDA) through MALLET (MAchine Learning for Language Toolkit) and Non Negative Matrix Factorization (NMF) through scikit-learn and how effective they are at varying levels of pre-processing. Through the collection of results in this specific case it can be said that LDA outperforms NMF at predicting topics while using low levels of pre-processing but both can be seen to even out when a higher level of pre-processing is used.

1 Introduction

When a human looks at a web page it is a trivial task to deduce what the content of the page relates to just by reading it. For a machine to do this however is a different thing entirely. A number of techniques for doing this have been developed which use algorithms to give meaning to words and distinguish between a group of semantic topics. In machine learning Topic Modelling is a method in which we build a statistical model to discover the different ‘topics’ that are present in a collection of documents. This is achieved by assuming that any given topic has particular tokens (words/phrases/names) that would occur only in that topic, or at least more frequently in that topic, than in any other. A simple example of this is if we are given two documents with the phrases “Referee and football” in the first and “Singer and guitar” in the second. We can clearly assume that the former is about football (from the tokens “Referee” and “football”), and the latter about music (from the tokens “Singer” and “guitar”) but we can also see that the token “and” is present in both documents. This is what can be called a stop word, i.e. a word that holds no definitive topic, “the” and “in” are also good examples of this. Most larger documents house a number of different topics at varying proportions so frequency of these tokens can become the deciding factor when attempting to evaluate the topic. In my project we will be using two different forms of Topic Modelling on 879 Irish blogs, which were nominated for Blog Awards Ireland in 2013, and seeing what accuracy can be achieved by each one while predicting their categories.

The primary topic model we will be using in this project is Latent Dirichlet Allocation (LDA) created by David Blei, Andrew Ng and Michael I. Jordan in 2002 which is a generalisation of Probabilistic Latent Semantic Indexing (PLSI), one of the earliest topic models created by Thomas Hofmann just 3 years earlier in 1999. We will be using MALLET (MACHINE Learning for Language Toolkit), which is an open source Toolkit. MALLET is programmed in Java and was developed by Andrew McCallum in 2002. [1]

The secondary topic model we will be using is called Non-Negative Matrix Factorization (NMF) which was created by Daniel D. Lee and H. Sebastian Seung in the late 1990’s. NMF was formerly known as Positive Matrix Factorization when a group of Finnish researchers were working on it in the mid 1990’s but Lee and Seung published two simple and useful algorithms for two types of factorizations after investigating the properties of the algorithm.

Later in this report we will go into more detail of how we plan to extract the text of an HTML document while trying to minimize loss of positive text (text that has a positive effect on deciding what category a specific topic belongs to) but also minimizing the amount of negative and redundant text (text that does not belong to any one category). We will also have a section in more detail on Topic Modelling, explaining the inner workings of LDA and NMF. We will then talk about my methodology and strategy as we went to implement my project and the comparisons that we could draw from the results we got and what we think could be done further using these methods.

2 Background Research

Background research was done to gain a full understanding of what level of text extraction is needed to be done so that as close to 100% of the positive text is kept and as close to 0% of the negative text is kept. Research was also done to find out the inner workings of topic modelling namely LDA and NMF.

2.1 Text Extraction

Most webpages seem structured and easy to read but once they are looked at in detail we can quickly see that in the code they are quite different. Figure 1 shows what a user sees when looking at a blog.



Figure 1. Users view of blog_0002.

Figure 2 below shows a section of code that represents just the text in Figure 1 above. The numbers on the left of Figure 2 indicate what line of code it is out of 8596 lines of code.

```
657 <h1 class='title'>
658 The Tale Of The Ale
659 </h1>
660 </div>
661 <div class='descriptionwrapper'>
662 <p class='description'><span>I write about beer, I drink beer, I brew beer and most
importantly, I simply enjoy beer. I'm a supporter of Irish Craft Beer and seek to
help raise awareness of Ireland's Independent Breweries. That doesn't stop me from
enjoying beer from all over the world of course.</span></p>
663 </div>
```

Figure 2. Source Code view of blog_0002.

When both pages are copy and pasted into a word count software it becomes truly obvious how important a good text extraction method will be. The content of this blog had a count of 10826 words whereas the code had 45476. Although this sounds like it may be a difficult process extracting text such as a newspapers story or in this specific case a blogs content from a page can be known as a very trivial process. McKeown et al. [2] claimed it could be done with just one simple heuristic. Time has progressed quite far from this claim however and as this time

progressed so has the complexity and style of the average web page. Each site generally has it's very own template and style and each page now has a host of different features such as scroll bars, navigation, advertisements, links to social media etc., this is even more true for blogs as most blogging sites are a form of social networking which encourage individuality through many customization options that have an effect on the blogs layout. Although the majority of the blogs we will look at in this project are from the same or very similar sites it is evident that they have different templates and styles, which will have a large effect on the text that we get back. For this reason it is to save time and increase accuracy that we will use an already existent method of text extraction by using a toolkit called Boilerpipe.

2.1.1 Boilerpipe API

Because we have decided to code our project in Java we will be using a text extraction API called Boilerpipe. Boilerpipe is a Java Library written by Christian Kohlschütter, which are based on concepts of this paper "Boilerpipe Detection using Shallow Text Features" [3]. The API offers numerous algorithms in which to extract text that affect the results.

Strategy	Description
ArticleExtractor	(default). Uses ArticleExtractor : A full-text extractor which is tuned towards news articles. In this scenario it achieves higher accuracy than DefaultExtractor.
DefaultExtractor	Uses DefaultExtractor : A quite generic full-text extractor, but usually not as good as ArticleExtractor.
LargestContentExtractor	Uses LargestContentExtractor : Like DefaultExtractor, but only keeps the largest content block. Good for non-article style texts with only one main content block.
KeepEverythingExtractor	Uses KeepEverythingExtractor : Treats everything as "content". Useful to track down SAX parsing errors.
CanolaExtractor	Uses CanolaExtractor : A full-text extractor trained on krdwrdr Canola . If you are curious :-)

Figure 3. List of Algorithms available for the Boilerpipe API

My initial thinking at this point is that although the *KeepEverythingExtractor* may return more negative text than every other option it will also be more likely to return more positive text. We also believe that a lot of the negative text i.e. links to other blogs may actually be a strength as they should hold words that are relative to the current blog. This is something that we will experiment with and talk about more in the progress section of my report. Boilerpipe also offers a number of options for output of which we will be using just plain text.

Output Format	Description
html	(default). Output the whole HTML document and highlight the extracted main content
htmlFragment	Output only those HTML fragments that are regarded main content
text	Output the extracted main content as plain text
json	Output the extracted main content as JSON. For details, see this page .
debug	Output debug information to understand how boilerpipe internally represents a document.

Figure 4. List of output options available for the Boilerpipe API

2.2 Topic Models

Topic models are based on the idea that, in a single document, we can assume there are a number of underlying topics, and in this document, we can also assume that words associated with a certain topic (relevant to the document) will appear more often than words associated with another topic (not relevant to the document). An algorithm can then be used to exploit this, meaning we can generate an observation of this method by using a machine. There are many different algorithms, such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF) and a few toolkits are available which when used, implement them. In the case of LDA we use MALLET and for NMF we will use scikit-learn.

2.2.1 Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model for collections of discrete data such as a corpus of text documents. It is one of the most common topic models currently being used and is the cornerstone of many other, more recent topic models. It was developed by David Blei, Andrew Ng, and Michael I. Jordan in 2002[4] and is a generalization of Probabilistic Latent Semantic Indexing (PLSI), created in 1999 by Thomas Hofman[5].

The general idea behind LDA is similar to that of your average topic model, that documents are represented as a random mixture of latent topics, where each topic is characterized by a distribution over words.[6]

2.2.1.1 LDA Model

There are a number of steps to LDA and firstly in the section we will talk in more general terms about what is happening and then go to show the more in depth workings of LDA.

We must go through every document and randomly assign each word to a topic from **K** (a predetermined number of topics). In each document **D** go through each word **W** in **D**. Using **W** we then take each topic **T** of **K** and calculate two things, 1) the proportion of words in **D** already assigned to **T** ($p(\mathbf{T}|\mathbf{D})$) and 2) the proportion of assignments to **T** over all **D** that come from **W**. $p(\mathbf{D}|\mathbf{T})$. Assign **W** to a new **T** where **T** is chosen by probability $p(\mathbf{T}|\mathbf{D}) * p(\mathbf{D}|\mathbf{T})$ or in simpler terms, the probability that **T** generated **W**.

1. Choose $\theta_i \sim \text{Dir}(\alpha)$.
2. Choose $\Phi_k \sim \text{Dir}(\beta)$
3. For each word position i, j in **D**...
 - a. Choose a topic $Z_{i,j} \sim \text{Multinomial}(\theta_i)$.
 - b. Choose a word $W_{i,j} \sim \text{Multinomial}(\Phi_{Z_{i,j}})$

α	The parameter of the Dirichlet before per document topic distribution.
β	The parameter of the Dirichlet before per topic word distribution.
θ_i	Topic Distribution for document i .
Φ_k	Word distribution for topic k .
Z_{ij}	Topic of the j th word in document i .
W_{ij}	The specific word.
M	Repeated choice of Document.
N	Repeated choice of topics and words within a document.
$\text{Dir}(i)$	Dirichlet Distribution for parameter i .

Table 1. Explanation of Symbols in Algorithm above and Figure 5 below.

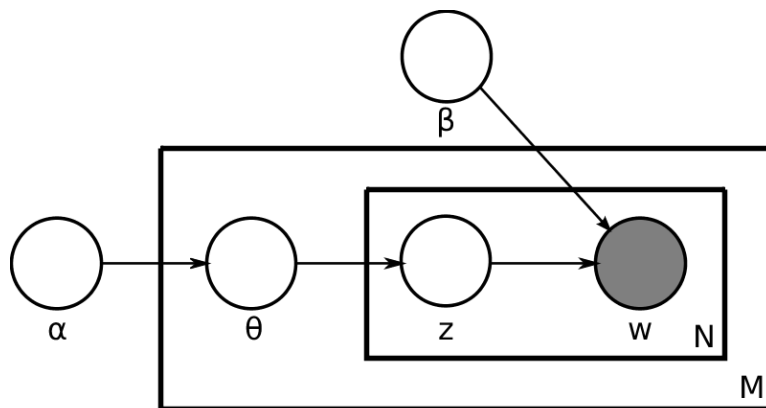


Figure 5. Plate Notation of the LDA model. [4]

2.2.1.2 MALLET

As we mentioned in the introduction MALLET is the tool we will be using to implement LDA. A number of customizations are offered by MALLET that allow the user to refine the results they are getting back, the most important of these in my case being a file called *stop words*. This file is full of a list of words that are removed from documents, which are imported from a directory. This is helpful as, in the case of blogs, there is a lot of redundant phrases and words such as “email”, “share on Facebook” which occur quite often and would have a large effect on the resulting topics. Although editing this file won’t change anything it is the basis of how we will add more stop words. To use MALLET there are 2 main commands that must be used from the *mallet* directory in command line or terminal. [7] We will talk more about these commands in the implementation section of this report.

1. Importing the directory.

- a. `./bin/mallet import-dir --input /users/username/database/ --output tutorial.mallet --keep-sequence --remove-stopwords`
- b. This imports a directory of text files (*import-dir --input path*) while keeping the original texts in order (*--keep-sequence*), strips out stop words (*--remove-stopwords*) and creates a *.mallet* file (*--output tutorial.mallet*) used in step 2.

2. Building a topic model.

- a. `bin\mallet train-topics --input tutorial.mallet --num-topics 20 --output-state topic-state.gz --output-topic-keys tutorial_keys.txt --output-doc-topics tutorial_compostion.txt`
- b. This opens the *tutorial.mallet* file from step 1 and trains MALLET to find k (*--num-topics k*) amount of topics. It then outputs a compressed *.gz* file that houses each word from step 1 along with the topic it belongs to (*--output-state-topic-state.gz*), a *.txt* file showing the top key words for each topic (*--output-topic-keys*) and finally a *.txt* file that shows the breakdown, by percentage, of each topic within each original text file imported in step 1 (*--output-doc-topics tutorial_compostion.txt*). The important thing to note about the key words file is that the words that appear here are in decsending order of importance to that specific topic.

2.2.2 Non-Negative Matrix Factorization (NMF)

Early work on NMF was done under the name of Positive Matrix Factorization in the mid 1990’s by a group of Finnish Researchers. It became popular after Lee and Seung did research on its properties and published some straightforward and effective algorithms for two types of factorization. [12] Although it is the method that we will be using for the second implementation on my project it has no strong statistical justification. It has however been implemented successfully in areas such as image and audio processing and text analysis. A paper justifying that NMF can be used as a main tool for topic modelling can be read here [15].

2.2.2.1 NMF Model

NMF is an unsupervised group of algorithms that perform dimension reduction and clustering side by side. It is a vector space method that gives a representation of data using only non–

negative constraints. These constraints can be used to show a parts-based representation of this original data because they only allow additive combinations of the original data.[8] An Initial matrix is expressed as an $n \times m$ matrix V , where each column is an n -dimensional non-negative vector of the original database. The standard NMF problem is to find two new reduced-dimensional matrices W and H whose combined product will approximate the matrix V or $V \approx WH$ [9]. Each column of W contains a basis vector while each column of H contains the weights needed to approximate the corresponding column in X using the basis from W . The dimensions of W and H are $n \times r$ and $r \times m$ respectively, where r is the number of document clusters. Assuming consistent precision, a reduction of storage is obtained whenever r , the number of basis vectors, satisfies $(n+m < nm)$. [10] Although NMF is primarily used for image analysis i.e. facial recognition; it has been shown to be useful for clustering on document collections, (see *Xu et al.*, 2003 [11] or *Pauca et al.*, 2004 [8]). We expect that, because NMF is not primarily used for topic modelling, it may fall a little short when compared to LDA.

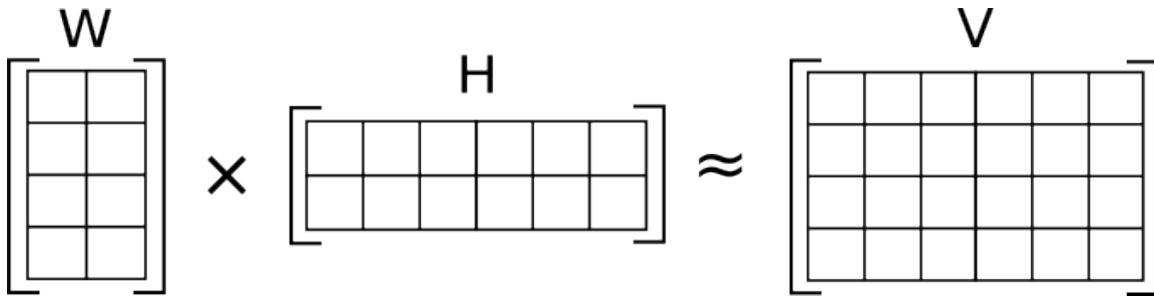


Figure 6. Breakdown of NMF

Figure 6 above is an example of how NMF breaks down a matrix V . Each row in V represents a blog and each column is a word within that text file. W is the composition matrix where each row in W represents a blog and each column represents a topics score when compared to that blog. H is the topic file and each row represents a topic and each column represents a key word within that topic. When W and H are multiplied they approximately reconstruct V .

2.2.2.2 scikit-learn

scikit-learn is an open source machine learning library for the Python Programming Language and was initially released in June 2007 by David Cournapeau. It offers a wide variety of simple tools for data mining and data analysis and has many different uses.[13] It is designed to incorporate with the Python libraries NumPy and SciPy. As with MALLET for LDA there are a number of inputs/outputs to be aware of. For equal comparison we use the same stopwords lists and blog files as done with MALLET. The number of topics will also be the same as done with MALLET. The scikit-learn code that we used initially only returned 10 key words in each topic at the start so we changed that to 20 to match MALLET. However, we will only be using the first 10 words to help judge later on. The code that we used later in the report was provided by my mentor Derek Greene and he also supplied some very informative slides on NMF. [14]

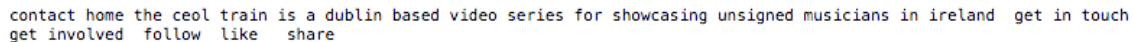
3 Implementation

In this section we talk my approach to the project and the strategies we used to gather my findings. We also talk about all the pre-processing we did before actually implementing my project fully and collecting the results. We do some implementation in this section but it is more in the sense of learning how and what to use to get results more than the results themselves.

3.1 Boilerpipe API

To begin with Boilerpipe we decided to run a few tests on the two algorithms that we felt would work the best, *ArticleExtractor* (the default/recommended one) and *KeepEverythingExtractor*.

When we first used *ArticleExtractor* and looked at the resulting text files we noticed that they seemed to be very bare compared to what had been on the blogs. A lot of what was left was words from the template of each page and not a lot of what the bloggers had actually written. Around 50% of the positive content (content that was categorisable) was left out in a large number of the blogs so we quickly had to decide against using it as it would give an unreliable set of documents for when we moved on to building a topic model. It also output a lot of blank text files when used, which meant that they would all need to be removed from the analysis. Figures 7 below shows just how little text can sometimes be extracted by *ArticleExtractor*.



```
contact home the ceol train is a dublin based video series for showcasing unsigned musicians in ireland get in touch
get involved follow like share
```

Figure 7. Output of *ArticleExtractor* on *blog_0057.html*

KeepEverythingExtractor however, would keep 100% of the positive content consistently. The negative of this method was that it would also keep quite a bit of negative content. When we realized this we took a deeper look into the negative content we were getting and realised that a large portion of the negative content was spread through three general things,

1. Links to similar blogs.
2. ‘Labels’ which were a list of keywords relevant to the topic of the blog.
3. Comments from users that were for the most part relevant to the blogs topic.

It was evident that it was a much better approach to use the *KeepEverythingExtractor* as the good seemed to out weigh the bad in its results. The main downside to using this algorithm was that it also output a number of blank text files when used (although less than *ArticleExtractor*). So that we could move on from this point and begin using MALLET we removed these 17 blogs from the group. Figure 8 below shows how much more text can be extracted by *keepEverythingExtractor*.

cover the led farmers ? crazy in love the stalks ? be there for the day the stalks ? let go shadows
 and dust ? gosling songs for aidan corner boy ? oxen of the sun the blood red mountain band ? falling for a
 broken heart folks and ghosts ? the king folks and ghosts ? ghost denis delaney ? not your enemy denis
 delaney ? the princess and the clown stevie cliff ? rooftops killer ceol ? don't mess with my rabbit the
 rattling kind ? spancill hill kasey smith ? big jet plane angus and julia stone cover the rattling kind ?
 open letters tristan carroll ? neon lights james mark donnelly ? constant state of mourning tristan
 carroll ? summer wicker bones ? the employee nella dwyer ? little child milky ? macho man rachael
 mccormack ? come back home rachael mccormack ? keep believing milky ? city heights ian o doherty ? there's
 fire on the mountain ian o doherty ? profit michael macleannan ? for you michael macleannan ? pen to paper
 keith grogan ? a thousand miles keith grogan ? 7 days morgan la faye ? ferp morgan la faye ? where is there
 rachel marie miller ? breathe the choir invisible ? hundreds and thousands the choir invisible ? your boyfriend
 mick mc loughlin ? i love it when kurtis murphy and rachel marie miller ? bulletproof stevie cliff- thorns the
 ceol train sessions directed and edited by ma kurtis murphy and rachel marie miller ? bulletproof the ceol train
 sessions video by mark doyle and t kasey smith ? daddy ? emeli sande cover the ceol train sessions kasey smith
 and paddy gur the ceol train sessions ? april 2013 the ceol train sessions t the stalks ? leave me alone the
 ceol train sessions directed and edited by ma future phantoms ? until i see the sun the ceol train sessions
 directed and edited by ma tristan carroll ? to be ready the ceol train sessions directed and edited by ma
 stevie cliff ? the only picture stevie cliff plays his ne ciaran lavery ? little more time the ceol train sessions
 video by mark doyle and t michael macleannan ? on the right side now the ceol train sessions video by mark doyle
 and t the blood red mountain band ? on these waters the ceol train sessions video by mark doyle and t mick
 mcloughlin ? little lion man the ceol train sessions shot and edited by mark d nella dwyer ? all about you the
 ceol train sessions shot and edited by mark d morgan la faye ? eyelids the ceol train sessions morgan la faye
 headlining greg clifford ? unfinished endings wicker bones ? must be santa cover wicker bones regale us wi
 morgan la faye ? happy happy christmas original song morgan la faye which som future phantoms ? let it snow
 cover future phantoms rocking t gavin james ? have yourself a merry little christmas the amazing talent this s
 aaron carroll ? silent night cover aaron carroll in the chur greg clifford ? let it snow cover gregs hits us
 with his ve james mark donnelly ? o holy night cover james belts out to fiach moriarty ? driving home for
 christmas cover the ever amazing fiach mo ciaran lavery ? till you're leaving el oso ? coming up easy
 kurtis murphy ? arizona kurtis murphy ? all i need is your love greg clifford ? take off your mask greg
 clifford ? changing everyday ciaran brennan ? every dog has its day ciaran brennan ? don't need nobody
 common wolf ? eleven ian o doherty ? i'd be thinking about leaving a great man with a great nella dwyer ? she
 moved through the fair cover the amazing nella dwyer g the ceol train sessions ? february 2013 the ceol train
 sessions t common wolf ? we come we go the ceol train sessions shot and edited by mark d san siro ? the tide
 san siro perform the tide ciaran brennan ? don't need nobody the ceol train sessions ciaran opens up the very
 greg clifford and band ? revolver the ceol train sessions greg clifford and band ro morgan la faye ? pound for
 pound a song to dance to from t ciaran lavery ? till you're leaving the amazing ciaran lavery ian o doherty ?
 temptation of eve the ceol train sessions video by mark doyle and t keith grogan ? seven days the ceol train
 sessions shot and edited by mark d milky ? neon shining the ceol training shot and edited by mark d common
 wolf ? lonely this christmas cover ho ho ho starting off th kasey smith ? merry christmas everyone cover kasey
 and myself play sha san siro ? merry christmas everybody cover here san siro cover merry the blood red mountain
 band ? suzies? farm the amazing the blood red mick mc loughlin ? this time aslan cover a top notch cover from th
 rachael mccormack ? keep believing here we have the lovely r the ceol train sessions ??? january 2013 the ceol train
 sessions t common wolf ? take it all the ever amazing common w corner boy ? go soft into the night the amazing
 corner boy in the ceol train sessions ??? march 2013 the ceol train sessions t mick mcloughlin ? fairytale of new
 york cover mick sings on a roof in t facebook twitter tags 1 all aboard 2 [gig videos](#) 3 [gig photos](#) 4 12 ceols
 of christmas website by vine design twitter facebook youtube back to top tags 1 all aboard 2 [gig videos](#) 3 [gig](#)
 photos 4 12 ceols of christmas

Figure 8. Output of KeepEverythingExtractor on blog_0057.html (Labels/Link Text in Green)

3.2 Generating the Stop lists

The first step that we felt was really important was learning how to use MALLET properly and to my advantage. We knew that a lot of pre-processing would need to be done in my project and that be able to use MALLET would make this pre-processing much easier. To begin with we would need to see how accurate MALLET was at predicting coherent topics. To do this we imported the directory of text documents into MALLET and using the resulting *.mallet* file built my first topic model, setting the number of topics to 20. There was a mixture of results from this that we will classify as *The Good*, *The Bad* and *The Ugly*.

- a) (55%) Topics 4, 6-10, 12, 14, 16, 18 and 19 were quite clear in what topic they belonged to and an example is shown in *figure 9* below which are all words in the Irish language.

16	0.01173 na agus ar ag le il seo bh de ir ach mar la os sa sin ch gaeilge mo
----	---

Figure 9. The 'Good'

- b) (10%) Topics 11 and 17 were close to a) above but also had a few vague words in their midst as demonstrated in *figure 10* below which is a topic on health that has a few obscure words in it such as “posted” and “Facebook”.

17 0.07279 cancer people chemo lovely health brain great hospital posted breast facebook life sick comments
depression mental twitter doctor kids

Figure 10. The 'Bad'

- c) (35%) Topics 0-3, 5, 13 and 15 were far too vague to associate with any topic and generally looked like *figure 11* below. Although it seems like a mostly social media themed topic that is only because these words occur in nearly every blog.

0 0.38729 blog posted comment ireland leave august reading wordpress follow post comments continue tagged email
posts july read irish photography

Figure 11. The 'Ugly'

To show the importance of removing certain words we ran a test on all the blog files which demonstrated the presence of the words "Facebook" and "twitter" and in how many different blogs they appear.

	<i>Facebook</i>	<i>Twitter</i>
<i>Number of blogs</i>	539 (61%)	557 (63%)

Table 2. Number of blogs containing a given token.

We realised at this point that a lot of these tokens appeared with common phrases such as "share to token", "like on token" and "token.com", where *token* can be changed to *Facebook* or *twitter*. We removed these new token phrases and ran again.

	<i>Facebook</i>	<i>Twitter</i>
<i>Number of blogs</i>	361 (41%)	371 (42%)

Table 3. Number of blogs containing a given token after "to token", "on token", and "token.com" were removed.

At this point we decided that removing the tokens 'Facebook', 'twitter', 'comment' etc., was a necessity in this case as they appeared in too many blogs. In the few blogs where the author may actually be having a technological discussion this could be a bad thing but it is the better decision because of the fact that technology blogs account for <15% of the blogs. Note that the tokens 'Facebook' and 'twitter' are just two of many that needed to be worked out like this. The results after this process was completed were much better than above and the percentages of *The Good*, *The Bad* and *The Ugly* were now 75%, 15% and 10% compared to 55%, 10% and 35% above. We knew that doing this manually, every time a topic seemed to have bad token, would result in far too much wasted time and effort and decided that coding a function to automatically do this was in my best interests. We could then add the results from this code to the pre-processing that MALLET does as it comes with a very easy fix for this in the form of a simple text file. One of the parameters when running mallet is "--remove-stopwords" where 'stopwords' is a text file

holding a collection of words that will be removed when training the topic, words such as “the”, “and”, “it” etc. As we mentioned earlier adding words to this file won’t actually have any effect on the results shown. Instead we needed to create a separate new file and include it when importing the text documents on the command line/terminal. To do this we added ‘*--extra-stopwords [filename]*’ to the command (shown below) when we imported the directory of documents.

```
bin/mallet import-dir --input ../Dropbox/FYP/workspace/TextExtractor/TextFiles --output
FYP.mallet --keep-sequence --remove-stopwords --extra-stopwords extra.txt
```

Now that we had the method of adding these extra tokens to a stopwords list we just needed to implement it. We wrote a basic file reading function which iterates through all the text files in the main directory and puts each unique word from each file into a hashmap of <String, Integer>. The hashmap would simply contain each unique word and a count of how many text files this word occurred in. When this is done we then iterate through the hashmap and write selected words to a file.

3.3 Choosing the Number of Topics

Now that this was done, and the most frequently occurring tokens were removed, we could look at the topics once more and see how much they had improved. We ran the MALLET commands again using the different stopwords files and a clear increase in consistency could be seen as it got closer to the 40% stopwords file.

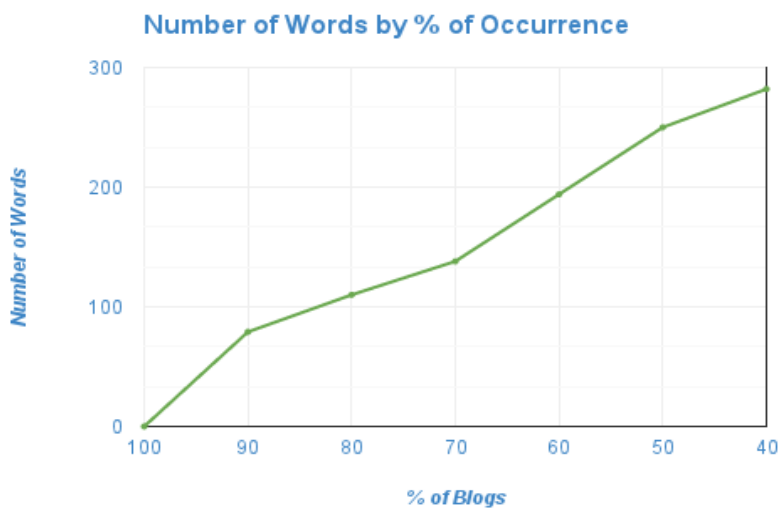


Figure 12. Number of words that occur in, at least, X % of blogs.

The reason we stopped at removing words that occur in over 40% of the blogs and didn't go lower was because it may then start removing the words that linked a topic to a category i.e. if one category was contained in 30% of the blogs it could affect this categories chances of being seen. When we had initially run MALLET above we could only categorise 14 of the 20 topics clearly when not including any extra stop lists. Now however, since we had cut down the number of blogs to 737, we could clearly categorise 14 of the topics. These cleared up even more when we began adding in the extra stoplists as Figure 13 below shows.

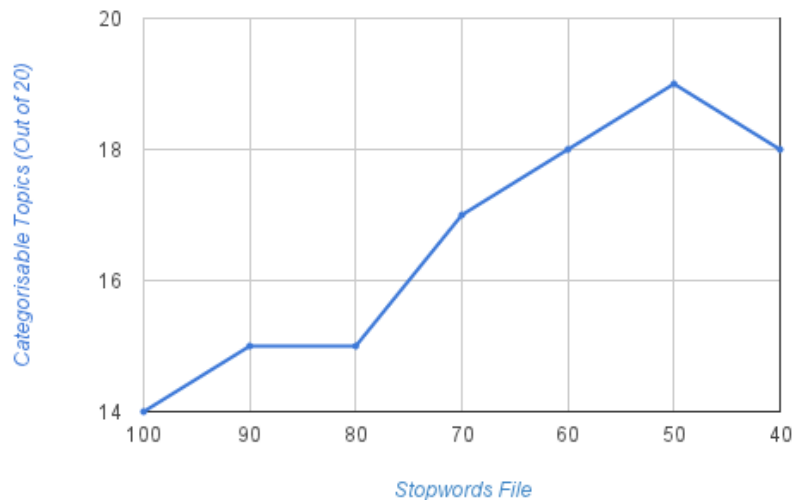


Figure 13. Categorisable Topics by Different Pre-Processing Levels

At this point in my project we also realized that there was another minor issue that needed dealing with before we could progress further. Choosing the number of topics to build a model with can be a difficult thing to do. The main reason it can be difficult, especially in this case, is because we as humans tend to be very complicated when it comes to being defined and these blogs were an extension of them. In the case of these blogs, when read through, it can be seen that they can be diluted with various categories. An example can be seen below, through two separate posts on blog_0049.



Figure 14. Example Post 1 - Blog_0049



Figure 15. Example Post 2 - Blog_0049

Although the human eye, with context, could see these are both categorisable as Humor a computer may easily classify the first as Political (“Fiana Fail”) or Health (“General Hospital”) and the second as Outdoors (“River”). This only elevates the problem as they both belong to the same blog. In some cases an approach is to use cross validation on the task at hand (e.g. information retrieval, text classification). Another method is to take a nonparametric Bayesian approach [16]. Luckily for me, at the beginning when we were given all the .HTML files we were also given a file that contained a list of the categories that the blogs could be assigned to and another list containing the mixture of categories each blog was assigned to. There were a total of 30 different categories in the collection.

Arts & Culture, Beauty/Fashion, Best Designed, Corporate, Craft, Diaspora, Eco/Green, Fashion, Food/Drink, Group, Health, Humour, Irish Language, Journalist, Lifestyle, Mobile, Music, Newcomer, News, Outdoors, Personal, Photography, Podcast, Political, Popculture, SME, Science/Education, Sport/Recreation, Technology, Video and Youth.

After we looked at these categories and the blogs within them we realized a few important things. *Best_Designed, Group, Humour, Journalist, Mobile, Newcomer, News, Podcast, Video and Youth* were all categories that would not reflect themselves but rather be a mix of the rest i.e. A News blog could be a few posts about a sports event but not be categorised in *Sport/Recreation* also. This could lead to a lot on innaccuracy, not because the topics were wrong but because the category assigned to the blog did not reflect it’s content. This left 20 categories. Narrowing down to just 20 categories did mean that we had to cull the blogs which were solely assigned to any of the 10 categories from the above list. This also altered the number of blogs being analyzed which dropped to 737.

Next we had to come up with a method for caetgorising the topics within a key file so that we may compare them to the actual categories of assigned to each blog. At first my idea was to collect the 3 or 5 highest scoring blogs for each topic and then to see which categories these belonged to. We would to use the composition file that would be output by MALLET after training the topics. As we talked about above, the composition file follows the following format with the last two columns repeatinf for each Topic:

Document ID	Document Path	Topic #	Topic Score
-------------	---------------	---------	-------------

If there was one or two categories that occurred in all of these then it would be safe to say that the topic in question belonged to those categories. When we thought about this more though we

realised that, for topics that were very vague and uncategorisable by eye, this method would still categorise them. This would be unwanted, as it would have a large chance to increase the accuracy of the results even though they were wrong. Another negative was that we would be using the composition file to try and predict the categories, when it is what we need to compare my categories to.

My second idea was something similar but a bit more refined. The primary step here is to look through all the given blogs and their actual category/categories and read in the ones that are only assigned to one category. With these read in we would have a large group of words that belonged specifically to each category. The next step would be to read in a MALLET keys file and get the keywords of each topic. Then we would cycle through every key word in every topic, and score it on two different things:

1. The number of times this key word occurs in each separate categories list of words.
2. The position of this key word in the topic as the key words are output in descending order of relevance to each topic.

A problem that arose from this last method was that 40% or more of the topics would get categorized as Diaspora everytime. As this didn't seem to be working we scrapped it and went for a much more basic approach. This approach would be to, when running the comparison program, prompt the user to identify each topic as a category. Although this has an element of human error and was not fully automated we felt it needed to be done this way.

3.4 LDA - MALLET

Now that we actually had gotten familiar with MALLET and all preliminaries were complete we could move onto implementing them properly. Firstly we would need to build a *mallet* file for each of the intervals we wished to build a topic model for. We did this using the import directory command via terminal on my computer:

```
./bin/mallet import-dir --input /users/sampower/dropbox/fyp/workspace/FYP/textfiles --output LDA100.mallet --keep-sequence --remove-stopwords.
```

The above command would output the LDA100.mallet into MALLET directory which didn't need an extra stopwords list. We repeated this command several times, changing LDA100 to LDA90/80/70/60/50/40 and also adding in the specified extra stop words list. Example for LDA40 below:

```
./bin/mallet import-dir --input /users/sampower/dropbox/fyp/workspace/FYP/textfiles --output LDA100.mallet --keep-sequence --remove-stopwords --extra-stopwords stoplists/extras/stopwordsCustom40.txt
```

The next step was use the .mallet files to actually build the topic model. We used the following command for each .mallet file produced by above. As mentioned in the research section this command would then output the topic keys and blog to topic composition file.

```
bin\mallet train-topics --input LDA100.mallet --num-topics 20 --output-state topic-state.gz --output-topic-keys LDA100_keys.txt --output-doc-topics LDA100_composition.txt
```

All that was left to do now was to co

3.5 NMF - scikit-learn

As mentioned in my background research on NMF we will be using scikit-learn. My mentor, Derek Greene, provided Python code that we could implement NMF with which saved a large amount of time. To build the key file all that needed to be done is change the output name of the file within the code and then run the file, with the directory of text files as an argument, as shown in Figure 16 below.

```
Sam-Powers-MacBook-Pro:nmf SamPower$ python nmf.py /Users/SamPower/Dropbox/FYP/Workspace/FYP/TextFiles/
```

Figure 16. Running NMF in Terminal

We did edit the code slightly to make it possible to read in an extra stop list file so that we could build key files the same as with MALLET. The next step in implementing NMF would be to build a composition file, something that we had to do ourselves. This wasn't a very complicated thing however as it was simply which topic matched each blog the most. This method allowed me to find the top 3 scoring topics for each blog, which were then written in the same format as the LDA composition file.

3.6 Scoring the Topics

Now that we had all the composition and key files we could finally assign categories to the topics and compare them using the composition file to the actual blog categories. As mentioned above we would show the user a list of categories and then show them each topic and ask them pick which category it belongs to based on the first 10 key words. Figure 17 below shows this in action. When comparing the prediction to the actual categories assigned we decided to just take the highest scoring predicted category and if it was one of the actual categories we would then say that it was a correct prediction, otherwise it was a false prediction.

```
List of categories...
[Arts & Culture, Beauty/Fashion, Music, Sport/Recreation, Personal, Eco/Green, Science/Education, Diaspora, Corporate, Photography, Lifestyle, SME, Craft, Outdoors, Food]
Which of the above categories do the following topics belong to.
Type NONE if you cannot decide.
Topic :0      0.13526 food recipe sugar cream butter recipes chocolate cake kitchen baking cook salt oil cooking chicken minutes bread water ingredients
Food/Drink
Topic :1      0.28389 business social media online page marketing video google company site click website information tips content awards support design advice
SME
Topic :2      0.04119 beer food taste ale brewery restaurant stephanie beers brewing quigley craft stout ipa france festival wine company cigar hops
Food/Drink
Topic :3      0.11227 st john history diamond mark sproule architecture century county street james william samuel tower park died castle sligo diamonds
Arts & Culture
Topic :4      0.07231 school education students science chemistry english class teaching history college student ty teachers learning teacher university labels st app
Science/Education
Topic :5      0.49415 photography wedding blogthis baby labels photos lovely pm friday december fun sunday weekend links photo amazing event events visit
Photography
Topic :6      0.17819 government party politics political state general british war property rights local public country minister labour practice society community tom
Political
Topic :7      0.31144 reading book continue writing books art tags categories tagged story short awards film festival review theme writers culture poetry
Arts & Culture
Topic :8      0.09629 weight food health tea training body eating loss fitness exercise healthy eat diet foods water brain depression pp goals
Health
Topic :9      0.0887 beauty skin hair review products makeup product ve tan blogthis nails eye nail labels face lip body brown cream
Beauty/Fashion
Topic :10     0.06662 cancer film chemo lovely movie breast sick hair folks movies fucking kids films friend disney star girl boob feeling
Health
Topic :11     0.06151 swim swimming water swimmers channel travel airport korea swimmer dover south aviation tokyo crew beach sea asia open japan
Outdoors
Topic :12     0.09986 game waterford team race munster season players league run half running football final rugby results donegal club road mile
Sport/Recreation
Topic :13     0.14509 fashion style dress wear outfit shoes shopping bag vintage black blogthis penneys hair white skirt street zara magazine jewellery
Beauty/Fashion
Topic :14     0.82943 ve ll didn left started happy hard hope head remember side months felt point bad mind fact live told
none
Topic :15     0.11465 music album aug festival band jul video live pm jun reviews theatre song performance track tour review gig songs
Music
Topic :16     0.01282 na agus ar le ag il seo bh de ir ach mar la os sin ch sa gaeilge mo
Irish Language
Topic :17     0.08372 google pinterest loading tumblr linkedin tagged digg stumbleupon kids reddit children print dog mum child baby parenting breastfeeding health
none
Topic :18     0.06168 craft date sewing fabric table wood pattern handmade shop labels jewellery etsy crafts knitting design pm bealtaine vintage crochet
Craft
Topic :19     0.07153 garden plants farm growing gardens plant gardening flowers powerscourt grow organic green vegetables trees sodshow flower spring tree vegetable
```

Figure 17. Manually Assigning Categories to Topics for LDA40 keys

4 Results

In this section we show the final results of predicting blog categories using LDA and NMF. We will also draw a comparison between them and try to evaluate why this difference might be there.

Words that occurred in more than X% of blogs removed	Percentage of Blogs Assigned Correct Category	Number of Blogs Assigned Correct Category
100%(Basic Stop list)	39.21%	291
90%	45.55%	338
80%	47.04%	349
70%	40.83%	303
60%	42.85%	318
50%	42.99%	319
40%	47.99%	356

Table 4. Results of LDA Topic Modelling.

Words that occurred in more than X% of blogs removed	Percentage of Blogs Assigned Correct Category	Number of Blogs Assigned Correct Category
100%(Basic Stop list)	19.81%	147
90%	25.61%	190
80%	24.29%	180
70%	33.02%	245
60%	44.87%	333
50%	46.77%	347
40%	41.51%	308

Table 5. Results of NMF Topic Modelling.

The main differences between LDA and NMF above are where there are low levels of pre processing. This would imply that LDA outperforms NMF when used in on large data sets when just using a very basic stop list. In Figure 18 below it becomes very evident just how much of a difference there is between the two during the early stages.

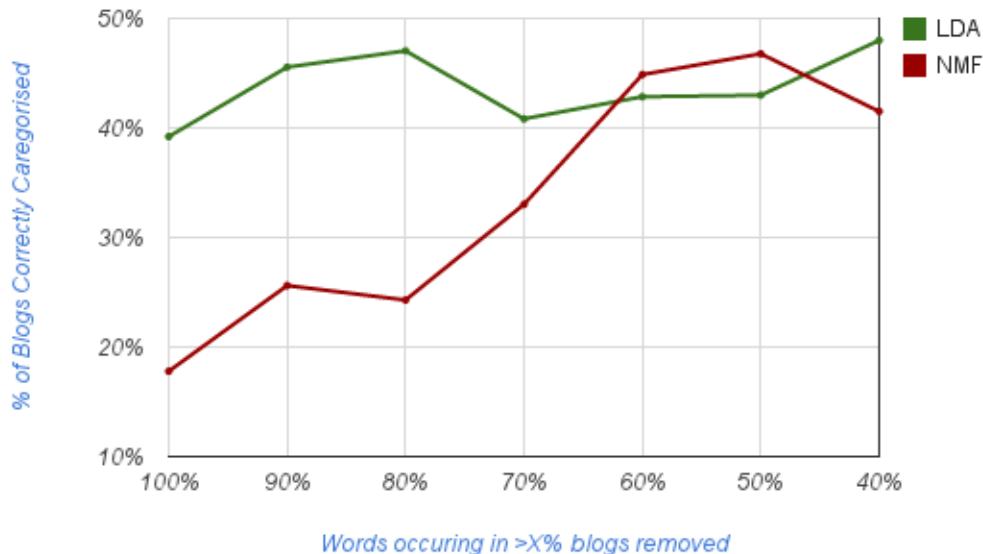


Figure 18. Comparing Accuracy of LDA & NMF Category Predictions

4.1 Conclusion

As mentioned above LDA outperformed NMF at lower levels of pre-processing but the results were about even when more pre-processing was added. The effectiveness of the method of text extraction that we used was key to building a good corpora of documents from which we could build a topic model. While the results did not go above the 50% mark we feel that they are satisfactory given the method of assigning categories and scoring that we used and the fact that we used blogs in our tests.

4.2 Future Work

If we had more time or were to hand this off project off to another party we would suggest implementing entity extraction to see how that improves upon the current results. Entity Extraction is a method where we recognise specific place and people names and assign them to a specific category. For example “Alex Ferguson” would be assigned to sport where as “Alex” and “Ferguson” on their own can be assigned to nearly any category.

5 Acknowledgments

I would like to thank my supervisor Padraig Cunningham and my mentor Derek Greene for all the time, guidance and help that they have given to me throughout the completion of this project. I would also like to thank the developers of Boilerpipe, MALLET and scikit-learn for their brilliant open source software.

6 Bibliography

- [1] McCallum, Andrew Kachites. MALLET: A Machine Learning for Language Toolkit. 2002 <http://mallet.cs.umass.edu>. [Accessed October 2nd 2013]
- [2] “Tracking and summarizing news on a daily basis with Columbia's Newsblaster.” - McKeown, K.R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Klavans, J.L., Nenkova, A., Sable, C., Schiffman, B. and Sigelman, S. HLT 2002. [Accessed November 19th 2013]
- [3] “Boilerplate Detection using Shallow Text Features” - Christian Kohlschütter, Peter Fankhauser, Wolfgang Nejdl. Available here [\[PDF\]](#) [Accessed November 19th 2013]
- [4] "Latent Dirichlet Allocation" - Blei, David M., Ng, Andrew Y., Jordan, Michael I, Lafferty, John (January 2003). Available here [\[PDF\]](#) [Accessed November 20th 2013]
- [5] “Probabilistic Latent Semantic Indexing” - Hofmann, Thomas (1999). (Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval. Available here [\[PDF\]](#) [Accessed November 20th 2013]
- [6] “Handbook of Mathematical Functions.” - M. Abramowitz and I. Stegun, editors. Dover, New York, 1970 [Accessed November 20th 2013]
- [7] “Getting Started with Topic Modelling and MALLET” – Shawn Graham, Scott Weingart, Ian Milligan. [\[Link to Tutorial Blog\]](#) [Accessed November 20th]
- [X] “Learning the parts of objects by non-negative matrix factorization” - Daniel D. Lee, H. Sebastian Seung [\[PDF\]](#) [Accessed December 1st]
- [8] “Mining using Non-Negative Matrix Factorizations” – V. Paul Pauca, Farial Shahnaz, Michael W. Berry, Robert J. Plemmons [\[PDF\]](#) [Accessed November 25th]
- [9] “Algorithms for Non-Negative Matrix Factorization” – Daniel D. Lee, H. Sebastian Seung [\[PDF\]](#) [Accessed December 1st]
- [10] “Comparative Analysis of Partial Occlusion Using Face Recognition Techniques” – N. Nallammal, Dr. V. Radha [\[PDF\]](#) [Accessed November 25th]
- [11] “Document Clustering Based on Non-Negative Matrix Factorization” – Wei Xu, Xin Liu, Yihong Gong [\[PDF\]](#) [Accessed November 25th]
- [12] scikit-learn examples [\[Webpage\]](#)
- [13] “Scikit-learn: Machine Learning in Python” – Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay [\[PDF\]](#) [Accessed January 21st]
- [14] “Matrix Factorization for Topic Models” – Derek Greene [\[PDF\]](#)

[15] “Learning Topic Models – Going Beyond SVD” – Sanjeev Arora, Rong Ge Ankur Moitra [\[PDF\]](#)

[16] “Topic Models” – David M. Blei, John D. Lafferty [\[PDF\]](#)