UNDERSTANDING DRACULA

Project report - LINMA2472 Algorithms in data science - UCLouvain

Lucien Ledune & Oscar Liessens - December 2018

Introduction

Bram Stoker's *Dracula* is universally recognized as a masterpiece and and a great influence on the horror genre. The clash between innocent Londoners and an ancient, malevolent entity dead set on spreading its evil curse indeed makes for an epic tale. In *Dracula*, the story unfolds as different characters meet with the vampire, and as some of them fall under his influence or gain suspicions about his true nature. Such a famous work surely has yet some secrets to reveal, so we decided to analyze it with different algorithms. We will begin by explaining somewhat the story and the characters for better understanding of our findings. Then, we will present the structure of this report and offer an overview of the different algorithms used.

The book is divided in 27 chapters, each only constituted of letters, journal entries and otherwise relevant documents, which serves to give an allure of reality to the novel as well as to offer the point of view of different characters. Therefore, it is worth noting that the story is mainly narrated by the central characters: Jonathan Harker, Mina Murray, Dr. Seward, Lucy Westenra, and the Pr. Van Helsing. The main story goes like this:

> "Jonathan Harker goes to Transylvania to help Count Dracula buy an estate in England. While he is at the castle a lot of strange things happen. His fiancée Mina and her friend Lucy exchange letters back home. Lucy starts acting weird and her friend Doctor Seward examines her. He can't figure out what's wrong and brings in his friend, a famous doctor, Van Helsing. Van Helsing eventually believes that Lucy has been bitten by a vampire. All the men in the book give Lucy their blood, but nothing works and eventually, after a lot of creepy behavior, she dies. They find out that she's undead, a vampire. They have to kill her by driving a stake through her heart. She is finally able to rest in peace. They realize that Dracula is behind all this and hunt him down. Mina falls under his spell and they hypnotize her so that they can track him down. This works and they finally catch up to him while he tries to go back to the castle. They kill him there and he turns to dust."
>
> - Source : http://schoolbytes.com/english/summary.php?id=371

To analyse this work, we have chosen four algorithms that could be of use. After defining the necessary amount of preprocessing, we will first show how we implemented two different tools, Latent Dirichlet Allocation and Non-negative Matrix Factorization, to extract topics from the novel, and discuss interpretations. Afterwards, we will detail our implementation of the PageRank algorithm with the goal of creating a network of connected characters. The third major algorithm implementation we will present is Apriori, which we applied on the book in order to extract sets of characters frequently associated in sentences, for every chapter. Finally, as some kind of bonus, we will present a functional simplified search engine designed to find the block of text a given query (probably) belongs to. Then, we will conclude by discussing our work and making remarks about ethical aspects in natural language processing and text mining.

(1) Text preprocessing

In order to be able to apply most algorithms to any kind of text, a preprocessing step is necessary. Throughout the report, we will use different kinds of division for the book, sometimes by blocks, sometimes by chapter, but the preprocessing step will always be needed and will always be the same.

We removed the parts before the first chapter and after the last one to only keep the book as data.

Raw text can be very difficult to interpret due to many factors, the most important being the fact that some words are basically the same but are written differently. For example, car and cars should be considered the same word in most cases, the same goes for Car and car.

First we are going to reduce every word to lowercase; this is self-explanatory. Then we will remove punctuation and most special characters, including eventual HTML tags (this one is done for reproducibility on other documents since our particular text doesn't contain any). Finally, with our text free from capital letters and punctuation, we can proceed the the stemming of words. Stemming is the process of reducing a word to its root form, so that similar words can be considered the same by our algorithms.

(2) Topic extraction

Topic extraction is a machine learning statistical model that aims to discover abstract topics in documents. We can easily interpret it, a document had a topic, and if it is from that topic it means that some words will be present in the said document more often than others. But a document can have multiple topics, the goal here is thus to find those topics.

Latent Dirichlet Allocation

Latent Dirichlet Allocation, often referred to as LDA, is a topic extraction technique based on a statistical model (probabilistic) almost equal to probabilistic latent semantic analysis that uses a Dirichlet distribution. This distribution helps to introduce the fact that a document only cover a small amount of topics and that a topic only contains a few frequent words.

LDA takes into input a bag of word matrix where rows are documents and columns are words. It then outputs two matrices :

- The document to topic matrix
- The topic to document matrix : It is often used to display topics by taking the n words with highest weights (= most frequent into that particular topic)

Non-negative matrix factorization

Non-negative matrix factorization is an algorithm based on linear algebra where we factorize a matrix into two matrices. All matrices must be non-negative, which explains the

name given to that algorithm. The non-negativity property makes interpretation more intuitive.

NMF also takes a bag of word matrix as input and outputs the same two matrices (same function but different results obviously).

Analysis

We are working on the book Dracula, so we are going to proceed this way :

- Preprocess the data as explained before
- Divide the document into chapters
- Apply LDA to our documents and then try to extract topics.
- Try to interpret results and maybe go further

For LDA (same goes for NMF), we need to specify the number of abstract topics we are looking for.

After a few tries, 3 topics seemed like a good option so we will use this one. For each topic we are using the topic to document matrix to display the words with highest weights in order to help us interpret the results.

```
Topic 0  : said come look time know hand shall came hels way night went like took van thing room ask man make
Topic 1  : said time come know shall look hels van mina day came good way hand luci night tell like count took
Topic 2  : said come look know time shall hand night van hels luci came went like good day thing man think room
```

Trying to interpret these results is still hard since they are very similar. In all topics it seems like people are talking (said), which is understandable given that our data is a book but they are so similar that we almost would want to extract only one topic from the book.

After these results we wanted to try something else, in order to get a better insight on the different topics of the book, so we are going to try LDA again, but this time we will use it on each chapter individually, extracting exactly one topic per chapter. This way we hopefully will capture some better information about how the different parts of the book are articulated. We will only show the 10 first topics in this report for a matter of space, but others can be seen with the code.

```
Topic  1 : driver hors look dark know time pass said road round
Topic  2 : count come know door room look place went hand said
Topic  3 : count room look great like said went thing time door
Topic  4 : count door room come window went look shall open letter
Topic  5 : tell dear know mina good love man say look told
Topic  6 : look old said luci like ye day lie come sea
Topic  7 : came men sea man mate pier night harbour sail ship
Topic  8 : luci night look came come time say window asleep sleep
Topic  9 : dear come know shall tell look luci ask night time
Topic  10 : said luci hels van sleep shall night good arthur come
```

This looks a lot better, we can now derive some information from the chapters, and we could imagine for example trying to guess what a given chapter is about based on these. Chapter 4 as an example could be about a letter, or chapter 5 about Mina (Dracula's loved one).

Another algorithm that is often used in topic extraction is NMF, we are now going to use NMF to compare the results with LDA.

```
Topic  1 : endur link hollow die twilight jump plain seiz scatter began
Topic  2 : couch cloth low sens doubt morriss wafer rank hous shadow
Topic  3 : cri sheer midst howl marri shut wind tops troubl dream
Topic  4 : clog die seven chang wood wild somewhat peril memorandum loud
Topic  5 : moved boy enemi final dark eye fallen leap nest exult
Topic  6 : materi place merci settl mad year doorway low crumbl sister
Topic  7 : sheer midst climb madam melt prey notic hear tri screech
Topic  8 : lull overcom lost carpathian constant trust wolv asid shook stenographi
Topic  9 : cours cheer kiss scatter strength leap left asid sunlight mouth
Topic  10 : round louder grown thing screech natur shriek figur approach centaur
```

One interesting thing to notice is that topics generated in chapters from NMF are a lot more diversified than LDA. Words with higher weights here are a bit more specific, making the understanding of the differences between chapters easier, yet a bit messier than LDA.

The results between the two techniques differ quite a lot. This might be explained by the Dirichlet distribution in LDA that assumes fewer frequent words.

(3) Graph representation of characters

Relation between characters

To approximate the relation between character, we are going to count the times where a character is mentioned in the same document as another one. So for example, if Dracula mentions Mina in the book, we will increment the relation [Dracula => Mina] of 1.

Given that our data is a whole book and not some well divided documents, we will need to divide it ourselves. But dividing by chapters here might seem like a very bad idea since chapters are very long and therefore a lot of characters will appear in every chapter.

To avoid this, we are going to divide the document in blocks of N lines. (here 10 lines).

The result is a matrix counting the frequency of two characters being mentioned together, this matrix will estimate the link between main characters.

For graph drawings, we will use a second matrix that is essentially the same but with the introduction of a threshold, where if a value is inferior to the threshold it is assigned 0 as value, this way we only draw pertinent links.

PageRank

Pagerank is originally an algorithm used by Google to generate a ranking between web pages. It measures the popularity of a web page but pagerank is not the only algorithm used in this case, it is just one indicator used to order the results of a research.

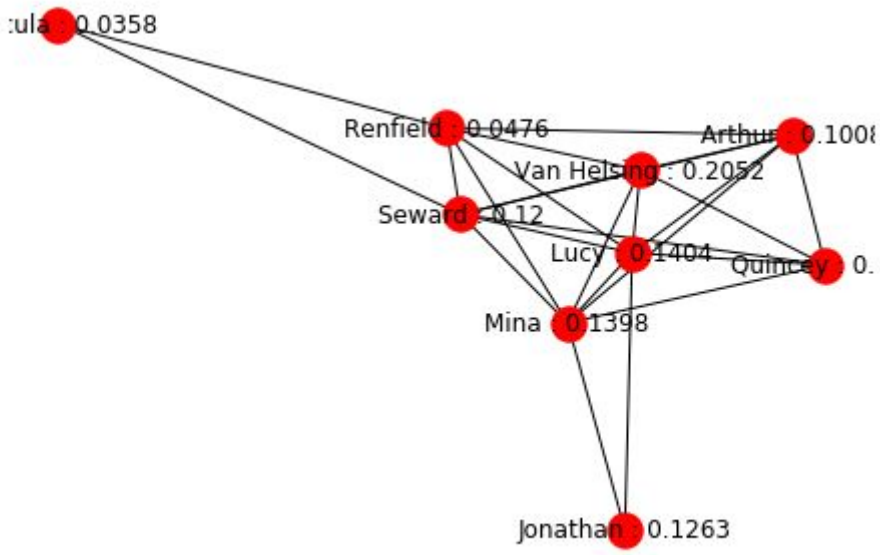
It works by assigning each page a score proportional to how many times an user would get on the said page by clicking on random links all pages.

Therefore a page is linked by a lot of other pages is going to have an high score. We can say that pagerank is a random walk on the graph. The output is a probability distribution of the likelihood that the user gets on a said page by randomly clicking.

The algorithm therefore calculates which pages are the most "central" ones, in our case it can be interpreted as which characters are the centrals ones, and are connected most to other main characters.

It means that we can use it to make a ranking of what are the most important characters throughout the story.

For PageRank, we will use the matrix without threshold, so that all links are included in the calculations.



(Upper left is Dracula and value for Quincey is 0.084)

Advantages: we can represent most important characters of the story very easily, it is a fast and iterative algorithm that converges to a solution.

Cons: As we can see, if a character is not directly linked to other characters it won't have a high score. This is easily explained by the fact that PageRank is based on the links between characters. But in our story Dracula is the main "bad guy" of the story and yet is ranked last... In our case it also means that if a character refers to another one as “my dear” or a similar formulation, the algorithm won’t take that into account.

Why? Well it is simply because he rarely appears in the book and therefore is not connected to a lot of other characters. So if a character is important but stays away from the other main characters it will not be ranked as high as he normally should be. We can add to that the fact that the author writing style

Although this gives us a good estimation on the written text itself, informing us that Dracula is not appearing a lot in the book. Without even reading the book, a lot of information can be extracted from this graph. First off, the main character isn't Dracula as the name of the book suggest, but the book is more about Harker and Van Helsing.

Second, Dracula is very absent from the book, he has some connections with Seward and Van Helsing (who is tracking him). But it doesn't seem like he is omnipresent.

(6) Frequent itemsets

Extracting frequent itemsets from a list of transactions is a very common objective in the data mining field. This process originated in basket market analysis. The goal was, with the past transactions of clients as a starting point, to assess which products are usually bought together. Here, we will try to extract sets of characters who are frequently found together in

a sentence of the novel. In order to get some sense of the evolution of these frequent character sets, we will do so for every chapter separately.

The main problem with frequent itemsets is that their mining takes more and more ressources as the number of items increase. The solution to this pickle is usually to use the Apriori algorithm, or one of its derivatives. Indeed, the premise of Apriori is that, for any frequent itemset A superset of itemset B, the itemset B is frequent too. The frequency of an itemset is defined as its support being greater than a given minimum support, the support being the number of transactions in which the itemset appears. Here, we will use Apriori to get a list of frequent character sets for every chapter.

Preprocessing

In order to apply Apriori on some transaction sets, we must first extract said transaction sets from our book. Our main goal, beginning to preprocess the data, was to obtain a list of character sets such as every character of a given set appears together in a sentence. To do so, we completed this list of tasks:

- Dividing the book in chapters
- Dividing every chapter in sentences
- Removing stopwords (obtained in the Natural Language Toolkit) and punctuation
- Dividing every sentence in words
- Keeping only words that were found in a set of character names we assembled :

```
["count", "dracula", "beast", "vampire", "jonathan", "mina",
"murray","arthur", "holwood", "quincey", "morris", "renfield",
"john", "seward", "abraham", "helsing", "lucy", "westenra"]
```

- Replacing every one of these character names by a single identifying name (e.g. : "count" -> "Count Dracula")
- Replacing sets of characters that consisted of the same character multiple times by sets containing only the character

Finally, we had our transaction sets ready. Practically, in python, this manifested in the form of a list of chapters, which each contains a list of character sets - themselves lists, as they contain names of characters.

Implementing Apriori

The next task, implementing a working Apriori algorithm, may seem trivial but can be quite tricky. Our algorithm takes as parameters a list of transactions (sets of names appearing together), an integer minimum support and an integer maximum length of character sets, and then proceeds as follow (for every chapter, as will be explained later):

- Creates a dictionary containing every character set and their support, if the latest is superior or equal to the minimum support
- Starts a loop that :
    - Stops if there are no more character sets in the level if the level has reached the maximum length. Here the sets are ranked according to their level, which is the number of character appearing in them.

    - Looks at every set and verify if it is of a superior level and if it is a subset of a frequent set
    - If it is, computes its support based on its subsets and adds it on the next level (if it's not already there)
- Returns a list of tuples containing two elements : a set of characters, and an integer representing the support of the set

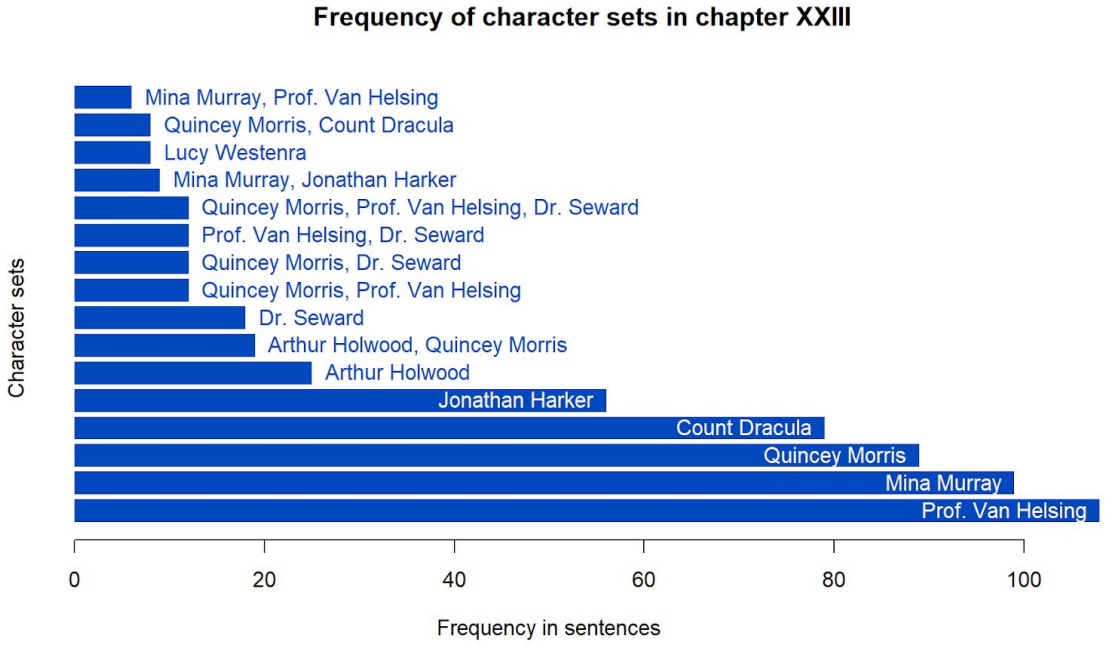
Applying the algorithm

Practically, our Apriori algorithm works with sets, and it seemed simpler to us to convert our character sets in codes (e.g. : "Count Dracula" -> "a") in order to create a list that can easily be taken as parameter. We thus created a function, specific to this project, that :

- Converted character names, strings, to identifying codes
- Applied Apriori on every chapter, and returned a list of "traditional" Apriori returns
- Converted character codes back to strings

The function worked just fine, and we were able to obtain the hoped returns: a list of chapter, each containing a list, each containing both a set of characters and the support of this set.

After that, we processed this return in such a way that it could be written to a csv file, and we used it to generate bar plots representing the frequencies of character sets in different chapter. As an example, see the figure below on which this process has been applied.



To interpret this, a bit of context. This chapter is constituted of two diary entries, Dr. Seward's and Jonathan Harker's, and it goes over two main events, a fight of the men of the story against Dracula and an interaction between Mina Murray and the Prof. Van Helsing.

The results seem coherent with the story. Indeed, as Van Helsing was present for both events of the chapter and in both diary entries without being their writer, it seems normal that he would be the most frequent character in this chapter. The same goes for Mina, who is a concern at the moment of the fight and participates in the second event of the chapter. It is no surprise either that Seward and Harker would be less well ranked as they wrote the entries constituting the chapter, and thus do not reference themselves very much. At last, it seems logical that Jonathan Harker is more frequent than Dr. Seward, because he was present for the first event which Seward narrated.

(5) Simplified search engine

To go further, we thought that we could use some matrix properties in order to create a simple search engine for the book.

Tf-idf cosines

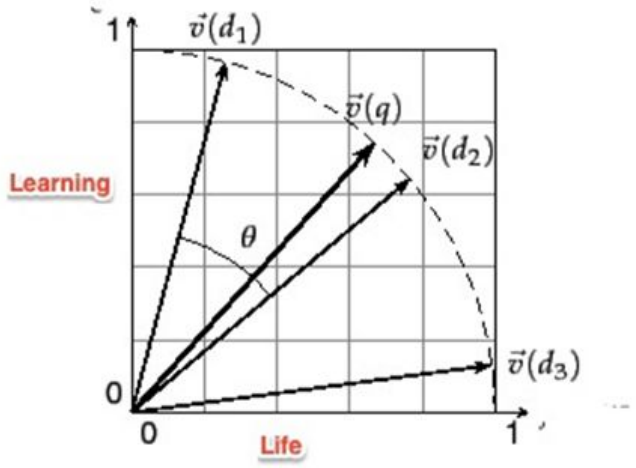
To create this section, we will use the tf-idf cosines similarity then return the highest value as result for the search, which is the most similar document according to the algorithm.

- Term Frequency (tf): Vector measuring the number of times a word appears in a document.
- Inverse Document Frequency (idf):

$$1 + \log(\text{Number of docs}/\text{Number of document with term in it})$$

The cosine similarity is then simply given by multiplying TF by IDF.

The goal here is to convert words into numbers that we can work with. Imagine a n-dimensional space with vectors for each documents representing their attributes. The document vector closest to query vector is going to be the most similar one. It means that the angle between them will be small, and as angle gets smaller, the cosines gets bigger. This is why the tf-idf cosines can be used as a measure to rank document by similarity to a query.



To try out our algorithm we are going to make a simple query that is related to the beginning of the book, hoping that it returns the first block of text as output.

Our query will be "I stopped at hotel royale, we had a good diner and i then visited a museum."

The algorithm outputs :

```
The block you are looking for should be block [1]

---------------------------------------
Block :   I¤¿CHAPTER I  JONATHAN HARKER'S JOURNAL  (_Kept in shorthand._)  _3 May. Bistritz._--Left Munich at 8:35 P. M., on
1st May, arriving at Vienna early next morning; should have arrived at 6:46, but train was an hour late. Buda-Pesth seems a
wonderful place, from the glimpse which I got of it from the train and the little I could walk through the streets. I feared
to go very far from the station, as we had arrived late and would start as near the correct time as possible. The impression
I had was that we were leaving the West and entering the East; the most western of splendid bridges over the Danube, which is
here of noble width and depth, took us among the traditions of Turkish rule. We left in pretty good time, and came after
nightfall to Klausenburgh. Here I stopped for the night at the Hotel Royale. I had for dinner, or rather supper, a chicken
done up some way with red pepper, which was very good but thirsty. (_Mem._, get recipe for Mina.) I asked the waiter, and he
said it was called "paprika hendl," and that, as it was a national dish, I should be able to get it anywhere along the
Carpathians. I found my smattering of German very useful here; indeed, I don't know how I should be able to get on without
it. Having had some time at my disposal when in London, I had visited the British Museum, and made search among the books and
maps in the library regarding Transylvania; it had struck me that some foreknowledge of the country could hardly fail to have
some importance in dealing with a nobleman of that country. I find that the district he named is in the extreme east of the
country, just on the borders of three states, Transylvania, Moldavia and Bukovina, in the midst of the Carpathian mountains;
one of the wildest and least known portions of Europe. I was not able to light on any map or work giving the exact locality
of the Castle Dracula, as there are no maps of this country as yet to compare with our own Ordnance Survey maps; but I found
that Bistritz, the post town named by Count Dracula, is a fairly well-known place. I shall enter here some of my notes, as
they may refresh my memory when I talk over my travels with Mina.
```

Which is exactly what we expected. The tf-idf cosines similarity is simple yet it proved to be quite efficient when it comes to search for some particular information in a book.

We can easily see the use of such algorithms, making the search into unread books much easier. However we can point a limitation (at least in our implementation): it doesn't take synonyms into account.

Conclusion

Our main takeaway would probably be that the eponym character, Dracula, is a rather elusive one. Despite being the main focus of the story, he is very often referred to in an

implicit manner and is not a main character. Indeed, he's called by many names and many of the central characters don't understand him to be a vampire, and a main cause of worry, until it's already late in the story. In our opinion, this could constitute one of the attractive traits of the novel; it's a puzzle, and every little piece of it helps paint a picture of a mysterious and evil entity, without ever giving too much information at once. This aspect of mystery only serves to make the ghost of the influence of Dracula more terrifying.

We also noticed that, the novel being written in an epistolary style, it is even harder to break down. Indeed, the narrator change at many points in the book and this makes it hard to reveal some continuity or development in the story. Moreover, that different narrators don't refer to the other characters in the same way. Despite there being many times where a simple diary entry turns into a detailed account of events, in a very classic novel-like manner, it is interesting to note that the writing style differ between two narrators. This may indicate that it would be of some interest to investigate the different entries separately, and linking their style, topics, frequent itemsets and graph representation to their author. This could give us better insight on the narrating characters and their perspective on the events.

At last, we note that maybe a better work could have been done with a better understanding of the novel. Even with some notion of the story and characters, our knowledge of *Dracula* remains superficial. Given that this work has been analyzed by experts many times, we think that scanning their findings and key ideas would be a good starting point in order to ask better research questions.

Ethical aspects

Given that Bram Stoker's *Dracula* falls into the public domain and that it's a fiction, there are virtually no ethical concerns here, but the same can not be said about the general applications of natural language processing. Indeed, we write an awful lot of text on the internet, be that on Twitter, Facebook, mail platforms, or search engines. These all constitute amazing source of informations, whether on current events or on their authors. We mentioned above that *Dracula*, as defining characteristic, is narrated mainly by its characters. This means that the processes one would develop to analyze their point of view are very similar to those one would use to extract information about Twitter users, for instance.

This kind of information is often either free of use, because tweets are public, or easily obtained due to security failures, but is very valuable in marketing campaigns. Worse, it can easily be used to gain influence on a people during elections, as has been demonstrated by the Cambridge Analytica scandal where information on Facebook customers had been illegally obtained. Natural language processing thus constitutes a threat to both the privacy and independence of online services users.

But for every ethical risk, there are benefits. Natural language processing is how we get technology to understand us better, and thus making it easily usable to more and more public. It's also an amazing tool to condense the huge quantity of information of the internet in a way that is readily understandable by humans, which in turn could mean more informational equality between those who can navigate the knowledge network and those who can't. One thing is sure, natural language processing is the future and it's already there

with its perils and its promises. Now it's up to us to see that if fulfills the later without making the world a place where we are all characters of *Dracula*.