UNDERSTANDING DRACULA

LUCIEN LEDUNE – OSCAR LIESSENS

INTRODUCTION

- Dracula : a masterpiece novel of the gothic horror genre
- Synopsis : a perfect couple, some soldiers and scientists, and a malevolent tourist
- An epistolary style
- Reasons for this choice

INTRODUCTION

- Text preprocessing
- Topic extraction:
  - LDA
  - NMF
- Relationships graph
- Frequent itemsets
- Bonus : a search engine
- Conclusion

TEXT PRE-PROCESSING

- Only keep chapters in the txt file. (Remove unnecessary parts before and after the book)
- Divisions : Chapters/N sentences
- Lowercase
- Remove punctuation and special characters
- Stemming of the words : reduce to its root form

TOPIC EXTRACTION

- First extract topics for all chapters
- Then extract one topic from each chapter, trying to get the differences in topics and see the story progression

TOPIC EXTRACTION : LDA

- Full text extraction :

```
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
```

- Chapters Extraction (LDA) :

```
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
```

First chapter example : Jonathan travels to meet Count Dracula, travels in a coach with a driver

TOPIC EXTRACTION : NMF

- Another algorithm often used for topix extraction is NMF, we use it to extract topics from chapters

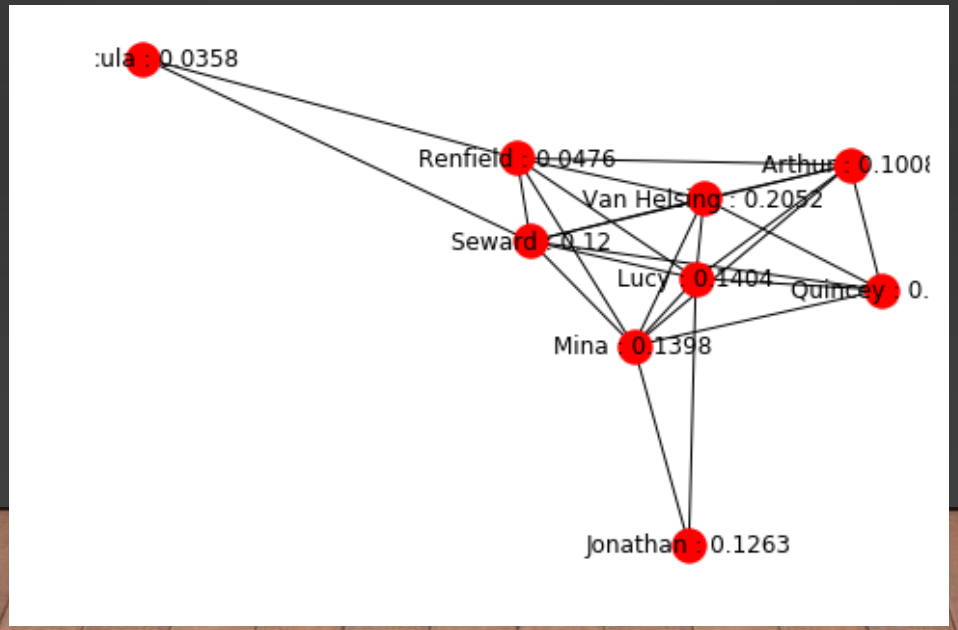
```
Topic  1 : endur link hollow die twilight jump plain seiz scatter began
Topic  2 : couch cloth low sens doubt morriss wafer rank hous shadow
Topic  3 : cri sheer midst howl marri shut wind tops troubl dream
Topic  4 : clog die seven chang wood wild somewhat peril memorandum loud
Topic  5 : moved 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
```

RELATIONSHIP GRAPHS

- We want to have a better understanding of the relationships between main characters in the book.
- We generate an adjacency matrix from the book, checking how many times characters are mentionned together.
- We use this to generate a graph of the book relationships
- Then use pagerank to determine a ranking between characters.

RELATIONSHIPS : PAGERANK

- Top left is Dracula, and value for Quincey is 0,084.
- Dracula is not the main character (rarely appears)
- Van helsing has the highest score.
- Limitation : In the book Dracula is often referred as different names (« the beast ... ») which makes it hard to capture every relation.
- Advantages : Fast iterating algorithm that enables us to rank characters.



FREQUENT ITEMSETS

PREPROCESSING

- 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

FREQUENT ITEMSETS

IMPLEMENTING APRIORI

- 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

FREQUENT ITEMSETS

APPLYING THE ALGORITHM

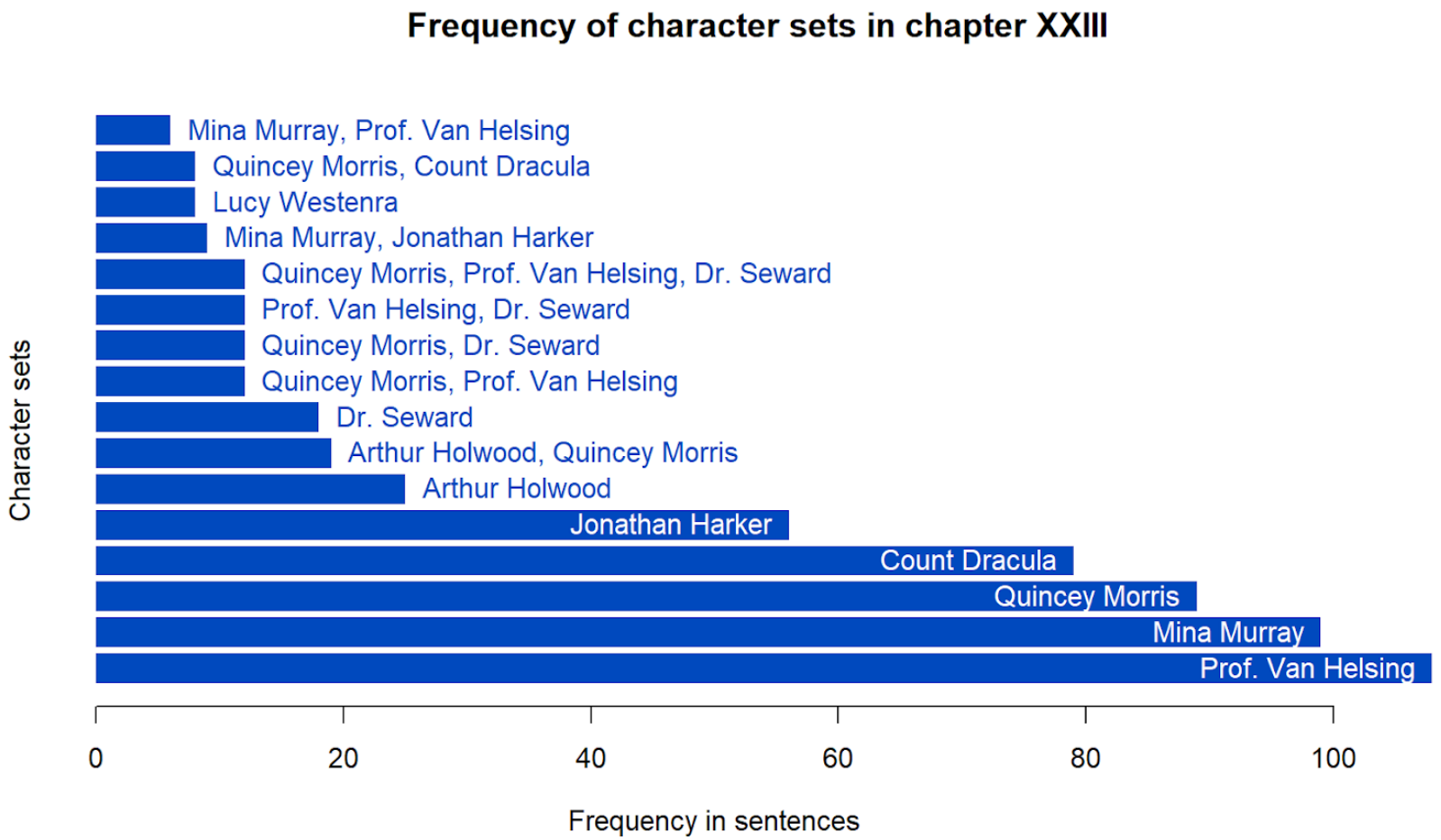
- Name sets in codes (e.g. : "Count Dracula" -> "a") in order to create a list that can easily be taken as parameter
- Data = multiple chapters !
- Function that :
  - Converted character names, strings, to identifying codes
  - Applied Apriori on every chapter, and returned a list of "traditional" Apriori returns
  - Converted character codes back to strings
- Returns : list > chapters > frequent itemsets > names set & support

FREQUENT ITEMSETS

APPLYING THE ALGORITHM

- Name sets in codes (e.g. : "Count Dracula" -> "a") in order to create a list that can easily be taken as parameter
- Data = multiple chapters !
- Function 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
- Returns : list > chapters > frequent itemsets > names set & support

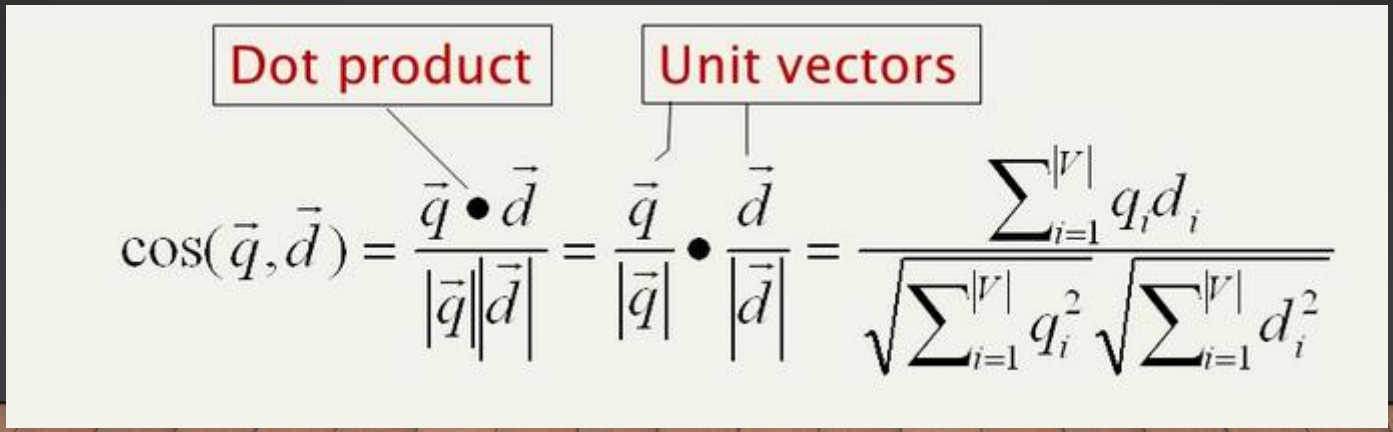
FREQUENT ITEMSETS



- Diaries :
  - Seward
  - Harker
- Events :
  - Men fight Dracula
  - Van Helsing talks with Mina

SIMPLIFIED SEARCH ENGINE

- Using tf-idf we tried to make a simple search engine for the book.
- This can prove very useful in case you havn't read the book and want to search for specific parts.
- Uses tfidf cosines to generate a similarity between query and documents, then return the highest one.



TF-IDF QUERY EXEMPLE

- "I stopped at hotel royale, we had a good diner and i then visited a museum.“
- This is about the very beginning of the book so it should return first block

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


---------------------------------------
Block :   ï»¿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.
```

CONCLUSION

- Dracula = mysterious character, elusive to text mining
- “Goal “of the novel : to paint his picture with not many informations
- Epistolary style :
  - It’s hard to find continuty or development because of changing POV
  - Investigate the novel by diary entries ?
- More knowledge of past analysis of *Dracula* would be helpful

CONCLUSION

ETHICAL ASPECTS

- No concern in this project, but in general natural language processing...
- Text on online services (Twitter etc) either:
  - Free of use
  - Made vulnerable by security failures
- Which can be used to:
  - Influence customers to buy products
  - Influence voters and change political landscape
- But there are benefits :
  - More access to technology
  - More informational equality

THANK YOU FOR YOUR ATTENTION ! :-)

ANY QUESTIONS ?