Technical report on extracting knowledge from online courses

1.The Source of Data

In order to ensure the quality of our information and to extract as much information as possible the source of data needed to have the following attributes:

- easily reachable
- certified quality
- well defined and persistent structure
- presence of data from various fields of study

All the above attributes represent constraints that need to be applied in order to discover a suitable data source. We started by considering all the known possible sources like, online encyclopedias (e.g. wikipedia), online courses web sites (e.g. coursera.org), university web sites (e.g. Open Yale course), online repositories (scribd.com), etc.

We consider a source to be easily reachable if we can grab the data through a common use protocol (e.g. FTP). Also this attributes refers to the format in which the data is stored. Preferable formats would be xml, json, text, rtf or any other type of format where text can be easily gathered. Extracting texts from PDF/PPT/Video or developing specialized web crawlers is also a good solution, which we will consider in the future, but the effort needed to develop such specialised software made us not to consider such sources for this experiment

Certified quality is very hard to obtain from online sources, it means that the people behind the data are certified by a recognised authority. This is not the case of wikipedia or other internet contributed web sites. The sources that satisfy such a restriction are university open courses web sites (e.g. MIT open courseware) or well known online courses web-sites (e.g. khanacademy.org).

Data may be represent in well defined structures like xml or json, but this condition alone is not enough. Let us take the example of coursera online courses, although they are easily reachable and have a certified quality, each course has it's own structure defined by the authors. In order to be able to analyse all the courses from coursera.org we would have to develop or adapt our tools for each particular course (numbering 1776 in January 2016). Thus persistence of the a well defined structure becomes important.

In order to make sure that our framework is universal we need to have a source that data from many field of study. A considerable part of the studied sources contained data related with the engineering field of study. But we want to incorporate fields like fine arts, humanities, social or sciences.

After applying all these constraints on our initial pool of sources we chose to use Open Yale Courses (OYC) web-site (http://oyc.yale.edu/). OYC are easily reachable, each course can be downloaded as a ZIP package. OYC is a prestigious university, in 2014 it was ranked as number 10 in the world[1], thus quality of the data obtained should more than adequate.

Probably the most important factor in choosing OYC as our data source is that all courses, no matter the field of study, have the same well defined structured. Each course contains:

- resources, usually PDF files of research papers that student must read.

[1] http://www.topuniversities.com/node/9201/ranking-details/world-university-rankings/2014

- sessions, html files for each lecture containing videos of the lectures.
- transcripts, html files with voice transcripts of the lectures.

Regarding the fields of study that are present in OYC we can say that 23 distinctive fields are present (January 2016) from Astronomy to African American Studies. Thus our universality constraint is met.

Finally, we chose to focus on the voice transcripts (VT) of the lectures represented as html, available in every course from OYC.

2. Preprocessing

The preprocessing step refers to parsing the given input (html files) and extracting the meaningful data. This step is necessary as html tags are not of interest to our study and may disturb our analysis. Also from the html extracted texts we need to keep those words that add the most information.

We chose to implement all our programs in Python programming language, as it is a widely used general programming language. Also python code is portable and can be run on all the major operating systems: Windows, Linux, OS X. Python is known for the readability of its code and for its scientific libraries (e.g. SciPy[2]), making it a good choice for academic research.

In order to parse the html files we used an open source library, BeautifulSoup[3], available in the Python programming language. For each VT file we extracted the texts from the html paragraphs (<p> html tag). Due the well defined and persistent structure of OYC all the paragraphs in VT files contained only transcripts of the lectures, thus no additional filtering had to be done.

Next we chose to use Part of Speech Tagging (POS) from the Natural Language Processing (NLP) research field in order to filter out words that bring no valuable information. We define valuable information in this context as words that define academic concepts, e.g. Object Oriented Programming from Computer Science or the concept of a Planet from Astronomy. Due to the nature of the English language as well as other languages, e.g. Romanian (latin language), concepts are encapsulated as nouns.

NLTK[4] (Natural Language Toolkit) Python library was used to extract POS from the previous preprocessed texts. At first we chose only to consider nouns from texts and remove all other POS. For example the text *The rocket is present in the hangar* will become *rocket hangar*. Thus only the concepts are extracted from the above phrase, making it simpler for a possible student to focus on just those two concepts. In our quest to get as much valuable information as possible we run experiments introducing other POS like: verbs, adverbs or adjectives. We found that verbs can give valuable information in contexts where the action is important, e.g. a particular novel or history event. It would be very hard for someone to pinpoint the tie between the concepts of *roman emperor* and *Gallia* without prior knowing that *Caesar* conquered (action) *Gallia.* However university courses like those on OYC speak of various events from various topics that are usually abstracted as concepts. For example, upon enunciating the concepts of *conqueror**,** Gallia* and *roman empire,* it is very easy to assume that *Caesar* plays a very important role in this context. Thus we observed that actions are also represented as concepts making the use of verbs redundant. Adjectives add valuable data to

[2] http://www.scipy.org/
[3] https://pypi.python.org/pypi/beautifulsoup4
[4] http://www.nltk.org/

nouns, e.g. *Who is the first human on the Moon.* Just considering the nouns we would get *human* and *Moon* which will point to a list of 12 astronauts. However due to the fact that information is usually time stamped we would easily find the answer to be *Neil Armstrong.* Although some information may be lost by only using nouns, we can extract the concepts that will eventually help in finding the correct answer.

Besides resuming to POS extraction of nouns we also decided to maintain some form of structure of the text by keeping the ending dot of a phrase. Thus ties between the concepts could be preserved, based on the supposition that concepts appearing in the same phrase are related. For example the phrases "*Christopher Columbus discovered America. Portugal paid for the expedition"* would become *Christopher Columbus America. Columbus Portugal expedition".*

NLTK library was also used in order for noun lemmatization, aiming to remove inflectional endings only and to return the base or dictionary form of a word. This was done to remove the differences from nouns like *novel* and *novels,* which represent the same concept.

At the end of this preprocessing step a paragraph from a VT lecture would become a text containing lemmatized nouns LN separated by dots.

3. Text to Graph

In order to keep the relationship between concepts that appear together often we had to use graphs as text representations. Graph theory is mature enough to allow us to allow us to obtain information out text based graphs.

Lemmatized nouns obtained from the preprocess step are represented as vertices. In order for two vertices to be linked together they must first be part of the same proposition. The second constraint is that they must appear one after the other, the first LN is joined with the next LN through an arc. Let us consider the text *Caesar defeated the gauls. "Caesar became roman emperor mainly because of its success in the war against the gauls".* After preprocessing the above it becomes "*caesar gaul. caesar roman emperor success war gaul".* Following the basic rules of constructing graphs from texts, stated earlier, we constructed the graph representation in Fig. 1.

As it can be seen in Fig. 1 the graph is directed, thus relations of precedence of concepts can be kept in graph format. The relation of precedence is important as it may imply causality/subdivision. Also the graph is weighted, thus if multiple occurrences of the lemmatized nouns are present (in the same position), the weight of the arc will represent the number of occurrences. The more occurrences the more important the entanglement between the concepts. Such may be the case that one concept can not be separated from the other in order to maintain the context's sense.

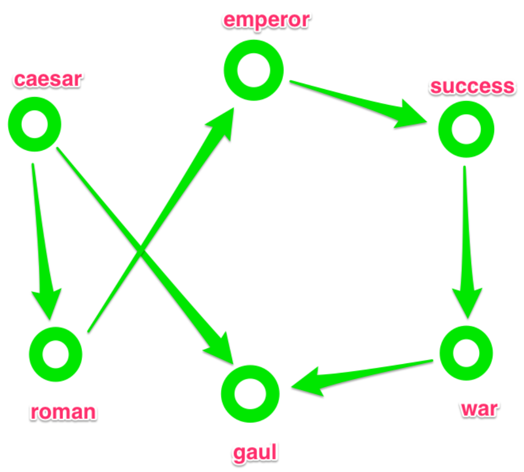

Fig.1 Representing a directed graph. Each node is represents a textual lemmatized noun.

4. Extracting information from the graph

For the current experiment we were interested in determining those concepts that are of the key building blocks in understanding a course/lecture/paragraph. The first supposition we considered was that those words appearing often in a text might be good candidates as keywords. But this idea is biased by the fact that some words appear often due to the language restrictions/necessities. Let us take the following example "*The first step is to build the aeroplane. The second step is to fly it and the third step is to land it safely*". As you can see the word *step* is the most often word encountered here, a common word that brings no real knowledge about the text, *aeroplane* would have been a better keyword.

A similar bias is found when trying to determine rank among web pages. Let us consider a web page as a vertice and the hyperlinks between various pages as arcs. Stating that a page is very important only by the number of links going outwards or inwards would not be entirely correct. Let us take an example of web page from a tabloid where we can find an article saying that CERN's[5] Large Hadron Collider (LHC) will create a giant black hole. This page may be shared on Facebook 10000 times. Let us consider another article posted on CERN's website stating that LHC will not end the world as we know it. CERN's website is not a mainstream one thus the number of links pointing to this page will be far less than 10000. Does this make the tabloid a better source of information?

Let us try another way to rank the pages presented in the above example. In ranking a page we don't only consider the number of links pointing to a particular website, their source is also important. The article from the tabloid will generally be shared by people that have no or very few expertise in physics. But the article from CERN will probably be shared and linked by experts, thus making CERN's website a more valuable. An algorithm that mimics to some degree the just mentioned idea is called PageRank. It was developed by Google's cofounders[6] and used in their search engine.

In this experiment we will use PageRank as way to determine key vertices in the graph constructed from lemmatized nouns extracted by us from OYC.

5. From keywords to useful learning materials

[5] European Organization for Nuclear Research

[6] Page, Lawrence, et al. "The PageRank citation ranking: bringing order to the web." (1999).

The goal of the current experiment is to create useful materials/formats based on which potential students can be guide on which concepts to focus, in order to understand the course/lecture. At the current state we can extract keywords from a course lecture or paragraph. However having a certain number of keywords is not very helpful as it might overwhelm a potential student with the number of keywords, not knowing which to learn first.

We considered developing two conceptual formats. The first format will guide the student from the first lecture to the last lecture by pinpointing the keywords as the course evolves, we name it *Tree of Knowledge* (TK). The second format should test the knowledge inherited by the student, we name it *Puzzle of Knowledge* (PK).

5.1. Tree of Knowledge (TK)

For the *Tree of Knowledge* we considered two granularity levels, lecture and paragraph. Basically keywords for an entire lecture or paragraph from a lecture. are extracted based on the methods mentioned before. Such a TK is depicted in Listing 1.

```
transcript06.html:star planet thing
--------1 paragraph: hot jupiters
--------2 paragraph: orbit thing
--------3 paragraph: orbit one
--------4 paragraph: planet kind
……………………………………..
--------61 paragraph: jupiters hot
--------62 paragraph: disc jupiter
--------63 paragraph: time problem
```

Listing 1. Example of a *Tree of Knowledge* extracted from lecture 6 of OYC's course Frontiers and Controversies in Astrophysics 2.

As you can see in Listing 1. on the first like the name of the lecture is stated together with 3 keywords. These keywords are representative for the entire lecture. We developed the program in such manner that a student can choose the number of keywords per entity. Thus he can choose as many keywords as he is able to absorb based on this experience with the domain taught. On the next few lines of Listing 1. you can observe that 2 keywords are depicted for each paragraph in the lecture. Again the number of keywords can be chosen by the student.

5.2. Puzzle of Knowledge

Out of the many forms for testing knowledge we chose a puzzle as the rules of playing are simple and implies interaction between the computer and human adhering to the idea of learning by playing. However the game of puzzle can become challenging in solving as the number of pieces grows. Thus we developed an algorithm to generate puzzle games of nXn (quadratic matrix) from the graph representations of texts that can be easily scaled in term of the number of pieces. The algorithm is depicted in Listing 3.

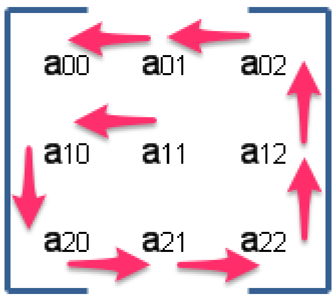

Figure 2. Passing through a 3x3 matrix in a spiral fashion. We start we ***a11*** and end with a00. The complete passing: a11, a10, a20, a21, a22, a12, a02, a01, a00;

We designed our puzzle to be matrices of nXn tiles. We decided to generate the puzzle pieces in a spiral starting from the center of the matrix, see Figure 2. The explanation for this type of passing is that we want to put the most important keywords in the middle of the puzzle. Thus the word that will be put in the middle of the puzzle is the the most important keyword. This positioning makes sense when we think that puzzles are usually built from the margins to the center, thus obliging the player to be aware of concepts that are gradually more important.

Let us take in consideration that after preprocessing and analysis we obtain a list of sorted (decreasing) words (LSW) by their PageRank coefficient, e.g. *caesar, rome, gaul, france, emperor, vercingetorix, legion, battle, roman.* Being 9 words we choose to generate a puzzle of 3x3, like in Figure. 2. First of all we place the most important keyword *caesar* in the middle on the matrix, position a11, and remove it from LSW. The next position where we need to place a keyword is a10. To determine the keyword in position a10 we need to check if has any neighbouring words already determined. Position a10 neighbors are a00 (north), a20 (south) and east (a11). Out of this possible neighbours only position a11 has been predetermined as *caesar.* Then from the graph representation of the text, see Figure 3 for a variant of such a graph, we get a list (LN) with all the neighbours of node *caesar (LN: rome, legion, gaul, vercingetorix).* According to LSW the word with the highest PageRank coefficient is *rome* from LN, thus *rome* will be placed in position a10. Further the algorithm will continue its spiral passing through the matrix and at each step it will chose from LN the word with the highest PageRank coefficient according to LSW.

```
  a00:roman a01:legion a02:battle
a10:rome a11:caesar a12:vercingetorix
  a20:emperor a21:gaul a22:france
```

Listing 2. Puzzle deduced from the graph represented in Figure 3.

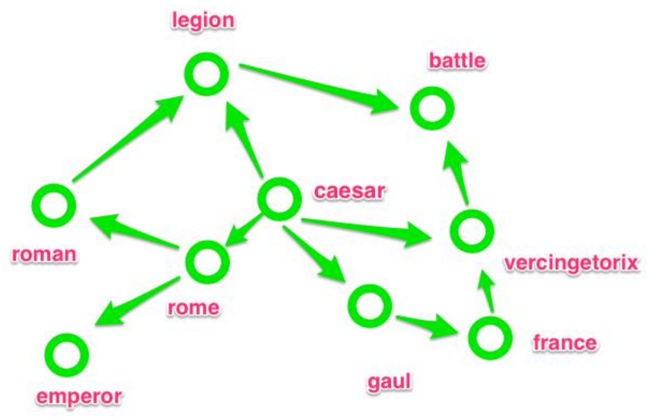

Figure 3. Possible representation of a text in a graph.

```
INPUT:
   ● graph G
   ● list of sorted words by PageRank coefficient
     calculated on graph G (LSW)
   ● vector matrix passing through the spiral method
     (VMP). Each element represents a position in the
     matrix (column and row)
   ● matrix M of nXn elements. Initially all elements have
     the value null
-aux=VMP[0]
-M[n/2,n/2]=aux
-VMP.pop(aux)
-for current_column and current_row in VMP:
     -LN=[]  empty list of words
     -if M[current_column+1,current_row] <> null:
          -aux=M[current_column+1,current_row]
          -aux2=G.neighbours(aux)
          -LN.append(aux2)
     -if M[current_column-1,current_row] <> null:
          -aux=M[current_column+1,current_row]
          -aux2=G.neighbours(aux)
          -LN.append(aux2)
     -if M[current_column,current_row+1] <> null:
          -aux=M[current_column+1,current_row]
          -aux2=G.neighbours(aux)
          -LN.append(aux2)
     -if M[current_column,current_row-1] <> null:
          -aux=M[current_column+1,current_row]
          -aux2=G.neighbours(aux)
          -LN.append(aux2)
```

```
-if ln <> null
      -sort LN decreasing according to each word’s
      PageRank on graph G
      -M[current_column,current_row]=LN[0]
```