Project Report On

Automatic Concept Map Generation from Text-based Learning Material

Centre for Educational Technology Courses
Indian Institute of Technology, Kharagpur
West Bengal - 721302

Guided By:
Prof. Plaban Kumar Bhowmick
Centre for Educational Technology Courses
IIT Kharagpur

Submitted By:
Bhiman Kumar Baghel (17CS60R74)
Ravi Rohan (17CS60R61)
Nikhil Agrawal (17CS60R70)

Index

Abstract

Natural Language Processing (NLP) techniques have been widely used to automatically extract information from unstructured text. In this report, an automatic concept map construction technique is proposed for the conversion of abstracted short texts into concept maps. The approach consists of identification of concepts from the text and finding relations between them. It uses the Stanford CoreNLP for anaphoric resolution. DBpedia Spotlight is used for the concepts identification and the relations between the concepts are extracting by Stanford OpenIE. It aims at extracting the propositions in the text to build a concept map automatically. The suggested propositions are relationships among the concepts which are not explicitly found in the paragraphs. This technique helps to stimulate personal reflection and generate new knowledge. Lastly, the map is visualized with the graph. The method provides users with the ability to convert scientific and short texts into a structured format which can be easily processed by the computer. Moreover, it provides knowledge workers with extra time to re-think their written text and to view their knowledge from another angle.

1. Introduction

Concept map is an effective tool to present summary of a learning material. Concept map is represented by a set of concepts and their relationships. They are different from formal ontology in the sense that concepts maps do not consider standard vocabulary to represent the relations. The concepts and relations are formed using free text.

Concept map for a given learning material is not unique. A concept map is represented as a graph G = (V,E) where V is the set of vertices representing different concepts and E is the set of edges representing the relations. An edge e ϵ E can be represented as a tuple (l,w) where l is a string that represents the relation name and w ϵ R represents the strength of the relation.

The problem statement is as follows:
Given a learning material (from varying domains)

- Identify important concepts that are discussed in the LM.
- Extract relations among the identified concepts.
- Determine weights of the extracted relations.
- Present a visual representation of extracted concept map.

1.1 Project flow

The project has the following phases:

a. Dataset Description
b. Data pre-processing
c. Coreference resolution
d. Important concept identification
e. Calculate concept similarity
f. Filter unrelated pairs
g. Extract relation
h. Visualize concept map

All these phases are briefly described in the next part of the report.

2. Concept Map

Concept maps were developed in 1972 in the course of Novak's research program at Cornell where he sought to follow and understand changes in children's knowledge of science (Novak & Musonda, 1991). During the course of this study the researchers interviewed many children, and they found it difficult to identify specific changes in the children's understanding of science concepts by examination of interview transcripts. This program was based on the learning psychology of David Ausubel (Ausubel, 1963, 1968; Ausubel, Novak, & Hanesian, 1978).

Concept maps are graphical tools for organizing and representing knowledge. They include concepts, which are usually enclosed in circles or rectangles, and relationships between concepts indicated by a unidirectional arrow from one concept to another. Words on the line, referred to linking words or linking phrases, specify the relationship between the two concepts. Based on Novak and Cañas (2006), a concept is defined as a perceived regularity in events or objects, or records of events or objects. A proposition is defined as statement about some objects or events in the universe, either naturally occurring or constructed. Propositions contain two or more concepts connected using linking words or phrases to form a meaningful statement.

3. Dataset description

The dataset consist of some information about any topic. It should be saved in ./data/ directory with any name you like but the format of the file should be '.txt'.

Example: ./data/Mydata.txt
This is the raw data file.

4. Data pre-processing

The raw data file is then pre-processed. The pre-processing is as follows:

- The raw data is separated into single line sentences.
- Sentences ending with '?' are removed.
- Special characters like '(', ')', '—', '-' are being replaced with '<whitespace>'.

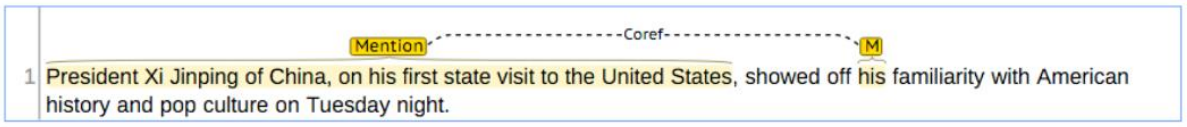
This process outputs a new '.txt' file in the ./data/ directory named as raw data file name followed by 'ext_data'.

Example: ./data/Mydataext_data.txt

The pre-processing is handled by the extract_data.py python program which is found in the ./code/ directory.

5. Coreference resolution

The pre-processed data file is then used as an input for co-reference resolution. In this phase the all expressions that refer to the same entity in a text are being found and then replaced with the corresponding entity.



Img.1: Coreference Ref. https://stanfordnlp.github.io/CoreNLP/

During this phase the pre-processed data file is taken as input by the Stanford CoreNLP API and generates an '.xml' file with the same name as pre-processed data file in the ./code/ directory, which contains all the expressions with their corresponding entities. This xml file is then parsed to extract all the expressions with their corresponding entities and a new file is generated with name as raw data file followed by '_coref' and the format as '.txt' in the ./code/ directory in which all the expressions in the pre-processed data are being replaced by the corresponding entities.

Example: ./code/Mydata_coref.txt

The coreference resolution is handled by the xml_parser.py python program which is found in the ./code/ directory.

6. Important concepts identification

The coreferenced data file is then used as an input for important concept abstraction phase. In this phase Spotlight API is being used for identifying important concepts. The Spotlight has two parameters i.e confidence and support which are being set as 0.4 and 20 respectively. The concepts are then stored in a file text file named as raw data file followed by 'concepts' in the ./code/ directory.

Example: ./codeMydataconpect.txt

The important concept identification phase is handled by a part of project.py python program which is found in the ./code/directory.

7. Calculate concept similarity

In this phase of the project, the concept data file is used as an input. Now for the comparison of two concepts, it is required to represent the concept in a vector. So, here the use of the FastText vector comes very handily. The pre-trained model of the FastText is already available on the internet, thus the input to the model is the concept text file. Its output is also the text file in which every line is starting with the word followed by the 300 floating values which are the vector representation of the word.

From this, the vector representation of all the words are generated, but one problem yet to be resolved to get the vectors for the concepts. As concepts can consist of more than one words and each word in concept has its own vector since FastText does not give the vector for the phrase. One solution is to take the corresponding sum of all the word vectors to make one vector for the concept, as taken in this project, or other option could be averaging all the vector. Now, the fixed size vectors for all the concepts are calculated, thus for the calculation of the similarity between the two concepts, the similarity between the vectors representing the two concepts is to be calculated. This similarity can be a cosine distance, Euclidean distance, etc. In this project, cosine similarity is used and a threshold the taken to separate related concepts and unrelated concepts.

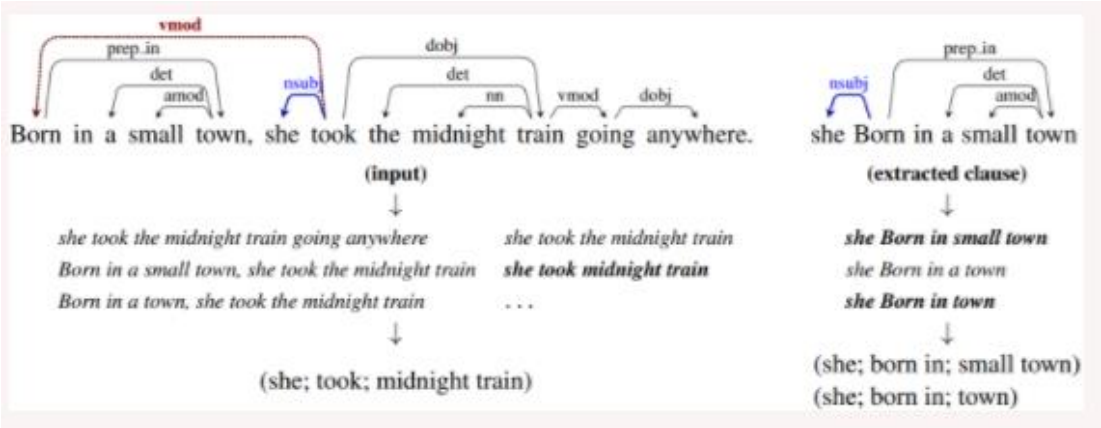
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Img.2: Cosine Similarity Ref. https://en.wikipedia.org/wiki/Cosine_similarity

where Ai and bi are components of vector A and B respectively.

8. Extract relations

In this phase of the project, the extraction of the relation between the concepts takes place which has the high similarity value than the threshold. So the extraction of relation from a sentence having subject and object of the sentence are the two concepts. Thus the head verb of the sentence becomes the relation between the two concepts directing from subject to object. In this project, the Stanford OpenIE API is used for the extraction of <subject; predicate; object > for the sentences where predicate becomes the relation between subject and object.



Img.3 OpenIE Ref. https://nlp.stanford.edu/software/openie.html

The input to the Stanford OpenIE is the referenced text file in which every line consist of a sentence and output of this is the text file containing the extracted relations finds by the OpenIE. Since many subjects and objects are not exactly matched to any concept in concept list, so the linking of the concept to the subject or object has to be done. For this, if any word in the subject or object is present in concept list, then the concept is linked to that subject or object. Also, the words at right or left to the linked word of the subject or object are added to the predicate at front or back respectively. Then the output of this is given to the graph module for the visualization.

9. Visualize concept map

Python API to visualize graph which uses graphviz to generate graph. In this phase the outputs generated from openIE after processing them will be fed to pygraphviz. OpenIE will generate a text file where each line will depict a relation between concepts.

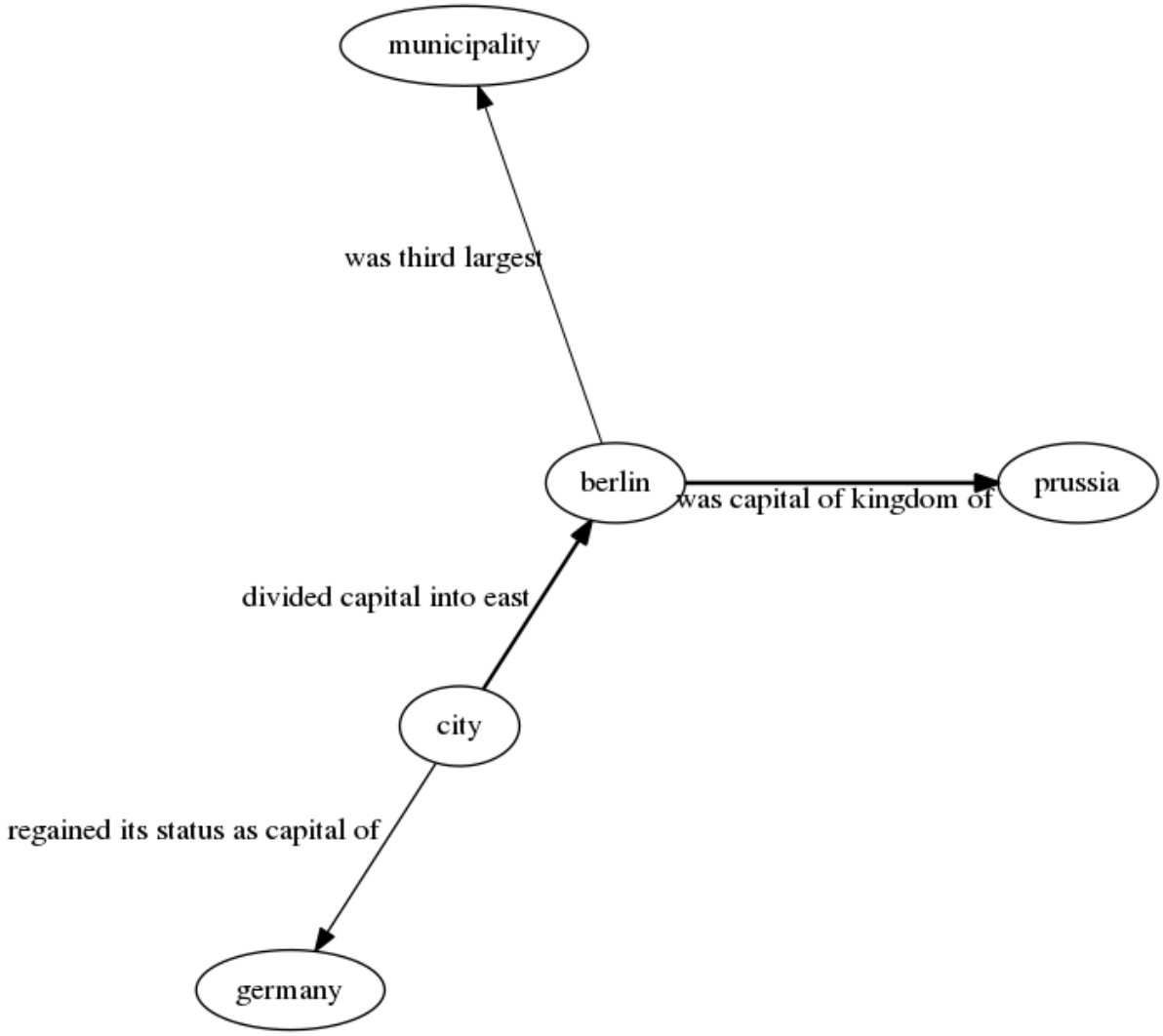
Pygraphviz will scan a line and add a node if it's not present and add a link whose width depends upon cosine similarity between concepts.

Node: Concepts.
Edge: If there any relation between pair of concept.
Edge_Label: Relation between those pair of concept.
Edge_strength: Edge thickness will depend upon relation strength which is function of concepts in vector notation.

Wil generate <raw_data>.dot file & then this will be fed to **Neato**(Engine). Explicitly define not to overlap nodes in graph by setting overlap = false in neato engine.

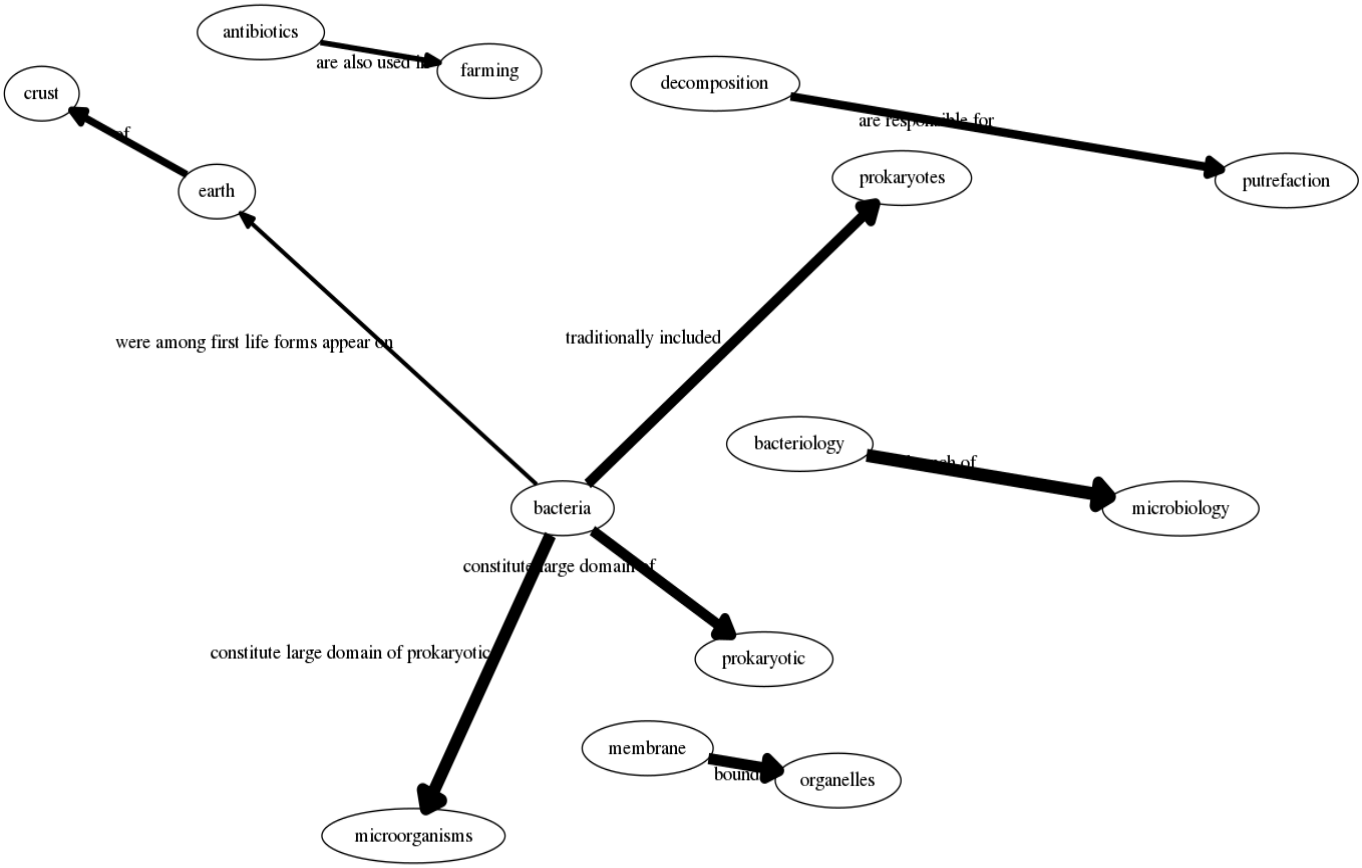
Input: <raw_data>gfile.txt.
Output: <raw_data>simple.png.

Example:



Img.4 Berlin Concept Map Dataset: Berlin.txt

Example:



Img.5 Bacteria Concept Map Dataset: Bacteria.txt

10. Observations

1. Pre-processing of text data improves performance significantly.
2. Simpler the text better the performance.
3. Highly dependent on Stanford OpenIE.
4. Lowering confidence on spotlight gives more unwanted concepts.
5. Stanford OpenIE mostly gives redundant relations.
6. The concepts extracted from spotlight and the relations given by Stanford OpenIE generally does not exactly match with each other.
7. Even provided with large dataset, the concept map generated is mostly sparse.

11. Conclusion and Future work

This report presents an architecture for constructing concept map from the short text written in the English language. This project uses well-known API like DBpedia Spotlight, Stanford OpenIE, Stanford CoreNLP, etc. which gives the noticeable results. The map generated is well structured as shown in the figures which is a huge success for the small project done with the help of APIs. The map can further be improved with the use of more NLP task like Rule-based reasoning (RBR) and Context-based reasoning (CBR) for anaphoric resolution. Evaluation can be carried out by calculating the degree of difference between the automatically generated outputs and the manually generated outputs by domain experts. Utilize link structure among Wikipedia concepts for extracting better relation between pair of concepts. Generate gold standard data for better evaluation and feedback it to system in order to improve performance.

12. References

[1] Pyspotlight: https://github.com/ubergrape/pyspotlight
[2] DBpedia Demo: http://demo.dbpedia-spotlight.org/
[3] Stanford CoreNLP: https://stanfordnlp.github.io/CoreNLP/
[4] Stanford OpenIE: https://nlp.stanford.edu/software/openie.html
[5] FastText: https://github.com/facebookresearch/fastText
[6] FastText Model:
https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md
[7] BeautifulSoup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[8] Pygraphviz: https://pygraphviz.github.io/