Concept Mapping

R&D Report

By

Ankith M S and **Sandip Ghoshal**,

Under the guidance of **Prof. Ganesh Ramakrishnan**

Department of Computer Science

Indian Institute of Technology, Bombay

Contents

1. INTRODUCTION

A **concept map** is a way of representing relationships between **concepts**. In a concept map, each concept is linked to another concept and gives visual representation of the hierarchy of the concepts .

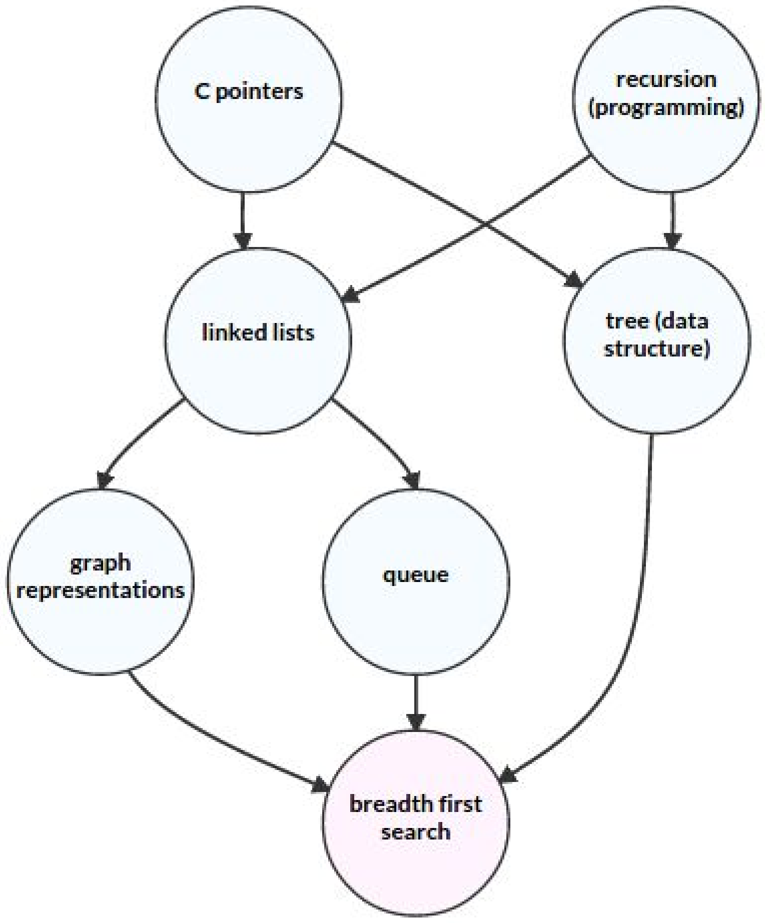


Figure 1: Concept map for breadth-first search (image courtesy: www.metacademy.org)

2. Training data creation

We used wikipedia and book corpus which are freely available from net as our training data . Since,majority of the ebooks will be in pdf format we needed to convert it to our required format

2.1. Pdf to text conversion

It takes pdf file as an input and returns the equivalent file in .txt format. There exist many tools which performs the above.

2.1.1 Tools used

For conversion from pdf to text we used the following tools:

i) PDF Miner

PDFMiner is a tool written in python for extracting information from PDF documents. Unlike other PDF-related tools, it focuses entirely on getting and analyzing text data. PDFMiner allows one to obtain the exact location of text in a page, as well as other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats (such as HTML). It has an extensible PDF parser that can be used for other purposes than text analysis.

ii) GATE (General Architecture for Text Engineering)

General Architecture for Text Engineering or GATE is a Java suite of tools originally developed at the University of Sheffield beginning in 1995 and now used worldwide by a wide community of scientists, companies, teachers and students for all sorts of natural language processing tasks, including information extraction in many languages. GATE has been compared to NLTK, R and RapidMiner.

GATE tools provides us tools for conversion from pdf to text indirectly. Using filters provided by GATE first convert from GATE to XML format and using another tool to convert from XML to TXT format .

In GATE there are few in-built application :

a. ANNIE
b. LingPipe
c. OpenNLP

Each of the above application has own set of filters some are :

a. Document reset PR
b. Language identifier
c. chunker
d. Tokenizer
e. Gazetteer
f. Sentence splitter

One can use any combination of the above set of filters to convert from PDF to XML.

iii) Zamzar

Zamzar is an online tool used to convert PDF to Text.

iv) PDFBOX

The Apache PDFBox library is an open source Java tool for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. Apache PDFBox also includes several command line utilities. Apache PDFBox is published under the Apache License v2.0.

We wrote a small snippet using Pdfbox library to convert from pdf to text

Conclusion :

Compared to all the above tools we got some satisfactory result with pdfbox . Since we used technical books , there were many equations and code snippets .Among the above only pdfbox was able to identify text, equations and code snippets as different.

2.2 Sentencification

Since pdf to txt conversion is noisy , we used NLTK tool for cleanup . Below is the sample output of our pdfbox tool

```
Symbolic logic is a mathematical model of deductivethought. Or
at least that was true originally; as withother branches of
mathematics it has grown beyond
the circumstances of its birth. Symbolic logic is a model
in much the same way that modern probability theory is a
model for situations involving chance and uncertainty.
```

After cleanup phrase, we wrote a python script to remove invalid break points. Last step of sentencification task, we used regular expression for breaking the paragraph into sentences .

2.3 Finding Entity Mentions

2.4 Preprocessing of Knowledge base

2.4.1 Lemmatizing

Lemmatization groups different inflected versions of words together into a base form for easy analysis.

Eg: HashMaps are implemented using a data structure known as a hash table .

Word : HashMaps
Lemma **:** HashMap

Word : implemented
Lemma **:** implement

2.4.2 Clustering of relations

we used Meta-academy to populate our relation database . We found many relations can be grouped together to form a single cluster .

2.4.2.1 Manual Clustering

We manually grouped the relations fetched from the meta-academy , gave a unique name to each cluster .

Relation Name	**Keywords**
algo_for	algorithm for, algorithm to, algorithms for
approx_to	approximation to
computed_using	to compute, computed using, computes, for computing the, way of computing

2.4.2.1 Word2Vec for autoclustering

Word2Vec is a google tool which gives similarity between two words . More information about word2vec is provided later chapter.

We trained word2vec with different values of vector length using our book - wikipedia corpus as training data and clustered relations having similarity score greater than 0.5 .

Relation Name	Keywords
algo_for	technique,method,algorithm
involve	involves ,requires,needed
typically	often,typically

3. Multi-R training

3.3 . Exploring features

Features plays a vital role in relation extraction . Better the features better the results. We found below are some of the features we can use it for relation extraction.

I. Entity strings
II. Entity type
III. head words of entities
IV. Entity level [Name,Nominal,Pronoun]
V. Bag of words and bigrams in Entities
VI. Bag of words or bigrams between two entities
VII. Syntactic structure
VIII. Dependency path
IX. Word Embeddings.

Consider the following example:
“***HashMaps are implemented using a data structure known as a hash table*** ”

Entity Strings :

Before we run relation extraction , we pass our data through entity recognition module, which tags the entities in the data . In entity strings feature entire entity content is passed as feature for our relation extraction module .

Entity String1 : HashMaps
Entity String2 : Hash table

Entity type :

Entity type is a feature on the abstract level , which takes care of , for certain relation some entity type pair doesn’t valid . For example : for **BornIn** relation entity type pair “**person**” and “**organization**” is not a valid.

Head words of Entities:

The head of a phrase is the element inside the phrase whose properties determine the distribution of that phrase. For example , if the entity is “long long integer” the head word will be integer .

Bag of words in entities :

To represent a bag-of-word feature, we can simply take a subgraph that contains a single node labeled with the token. Because the node also has an argument tag, we can distinguish between argument word and non-argument word .

Bag of words for Entity1 : HashMaps

Bag of words for Entity2 : Hash Table,Hash,Table

Bag of words or bigram feature between Entity Pairs:

A bigram feature can be represented by a subgraph consisting of two connected nodes from the sequence representation, where each node is labeled with the token.

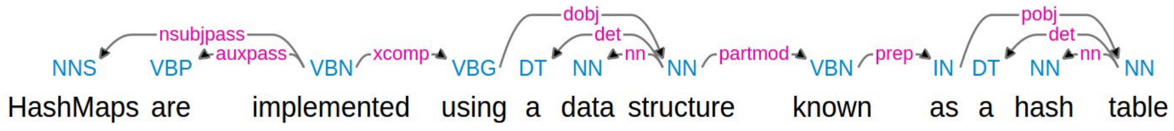
Bigram feature between entity pairs: {implemented using, using data,data structure}

Syntactic structure :

The main syntactic features are based on the full syntactic parse of the sentence . Example : HashMaps|NNP are|VBP implemented|VBN using|VBG a|DT data|NN structure|NN known|VBN as|IN a|DT hash|NN table|NN

Dependency path:

This plays a vital rule in rule based relation extraction . In simple rule based relation extraction system,the relation between two entities is the head of node connecting the given entity pair in the dependency path .



Word Embeddings :

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary (and possibly phrases thereof) are mapped to vectors of real numbers in a low dimensional space, relative to the vocabulary size . Detailed explanation how word embedding can be used in relation extraction is explained in word embedding chapter .

4. Multi-r testing :

4.1 Testing data

Once the training was done . In order to test the system we created the testing data by manual search and semi-automatic search . For semi-automatic search, we wrote a small crawler script which crawls the google search results and returns sentences having given relation name .

Overall we were able to collect around 300 testing data . Few of them are :

priority queue	head	implemented_as	A priority queue can be implemented as a heap
Monte Carlo experiment	PI	approx_to	I am doing a Monte Carlo experiment to calculate an approximation of PI.
BFS	depth of a tree	computed_using	Can we compute the depth of a tree using BFS or DFS algorithm?
DFS algorithm	depth of a tree	computed_using	DFS algorihm is one way of computing the depth of a tree.

5. Information retrieval :

Information retrieval is the process of fetching documents within the large collection of documents that satisfies the give the given queries . Since fetching all documents is redundant in many scenarios , one may be interested only in top-k documents . In top-k system , queries are evaluated using two major families of algorithms : 1) Term at a time(TAAT) and 2) Document at a time (DAAT)

5.1 DAAT

Document-at-a-time (DAAT) strategies evaluate the contributions of every query term with respect to a single document before moving to the next document.For a large corpora, DAAT has two two advantages

- DAAT implementations require a smaller run-time memory
- DAAT exploit I/O parallelism more effectively by traversing postings lists on different disk drives simultaneously

Both TAAT and DAAT can optimized significantly by compromising the requirement that all document scores are complete and accurate . An important optimization technique for DAAT strategies is termed max-score by Turtle and Flood. Given a recall parameter n, it operates by keeping track of the top n scoring documents seen so far. Evaluation of a particular document is terminated as soon as it is clear that this document will not place in the top n .

One of the optimization technique used in DAAT is explained by Andrei Z. Broder as follows .

5.1.1 Efficient Query Evaluation using a Two-Level Retrieval Process

In this method, queries are evaluated at two levels of granularity . At first level, it tries to find the candidate documents using a preliminary evaluation taking into account only a partial information . Once all candidates sets are identified full evaluation is performed on this list . Since for large systems full evaluation is an expensive task . so intention of two level retrieval process is to minimize this number of full evaluation .

Their approach allows both safe optimization, where no false-negative errors and returns top-k documents in the correct order and approximate optimization , some false-negative errors occurs at the risk of missing some candidate documents whose correct scores would have placed them in the returned document set.

Score for each document is calculated using

$$Score(d, q) = \sum_{t \in q \bigcap d} \alpha_t w(t, d)$$

α_t is a function of the number of occurrences of t in the query, multiplied by the inverse document frequency of t in the index .w(t, d) is a function of the term frequency (tf) of t in d, divided by the document length |d|.

Each term is associated with an upper bound term bound on its maximal contribution to any document score, UB_t such that

$$UB_t \geq \alpha_t max(w(t, d_1), w(t, d_2)......w(t, d_n))$$

Upper bound on the document's query dependent score is calculated by

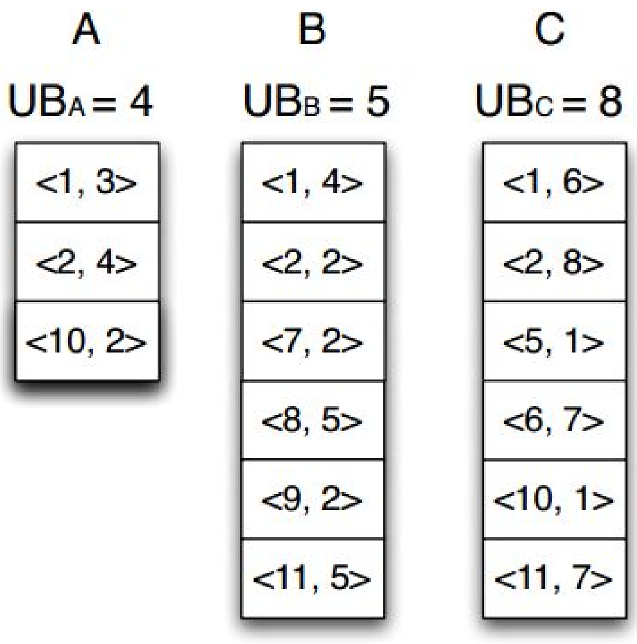
$$UB(d, q) = \sum_{t \in q \bigcap d} UB_t$$

The heart of this algorithm is the WAND operator which calculates the preliminary score of each document .

$wand(X_1, UB_1, X_2, UB_2, ..., X_t, UB_t, \theta)$ is true iff

$$\sum_{1 \leq i \leq k} x_i UB_i \geq \theta$$

Where x_i is an indicator variable for the presence of query term i in document d . If WAND evaluates to true, then the document d undergoes a full evaluation. The threshold θ is set dynamically y by the algorithm based on the minimum score m among the top n results found so far, where n is the number of requested documents. Thus in practice we will set θ = F · m where F is a threshold factor chosen . Higher the threshold less the documents will be fully evaluated .
How wand works are illustrated in below example . Let us consider the following query Q= {A,B,C} with all query weights 1 .



Above figure shows the posting lists and upper bound for each query term . For simplicity we consider k=2 i.e we want to fetch top two documents which satisfy the query.

After processing document 1 and 2

Heap	
docid	*score*(d, Q)
1	13 (θ)
2	14

Next step is to sort the cursor by their docid i.e after processing doc 1 and 2 , cursor of query terms A,B,C points to 10,7.5 respectively . So after sort we have

	C	B	A
p	1	2	3
docid	5	7	10

Next step is to select pivot . For p=1 we have

$$UB_c = 8 < \theta = 13$$

For p=2, we have

$$UB_c + UB_B = 8 + 5 = 13 < \theta = 13$$

For p=3, we have

$$UB_c + UB_B + UB_A = 8 + 5 + 2 = 15 > \theta = 13$$

The pivot is then set to term A (p = 3) and the pivot document is docid = 10. This means that the minimum docid that can potentially be in the top-k results is document 10. Next step is to move one of the query terms to 10 in order to continue processing. When compared to the naive DAAT algorithm, it is clear that WAND may reduce both the index access and the scoring costs. In this simple example docids 6, 8 and 9 are completely skipped in postings list for terms B and C and are not scored.

Importance of WAND Threshold :

- To handle mandatory terms,set to some huge value, H
- If we want to fetch only those documents which contains all query terms, set threshold to sum of all upper bounds .