CS6370 Course Project
Text Based Search Engine

Aditya B —— EE18B101 and Navnithan R —— EP18B009

Indian Institute of Technology Madras, Chennai TN 600036, India
ee18b101@smail.iitm.ac.in, ep18b009@smail.iitm.ac.in
https://www.iitm.ac.in/

1 Introduction

With the rapid development of the Internet, millions of bytes of data is being uploaded onto the world wide web on a daily basis. With such an information boom, there is a need for a reliable way to organize and retrieve information. The core of Information Retrieval is Text-Based Search Engines.

In traditional Vector Based Models for Information Retrieval, the IR system does not account for relations between words and fails to retrieve proper results in cases of synonymy, polysemy, etc. In this paper, we propose to use a IR System based on Explicit Semantic Analysis (ESA). Explicit Semantic Analysis (ESA) represents meaning in a high-dimensional space of concepts, automatically derived from human-built repositories such as Wikipedia thus doing away with the issues of not representing word relatedness [1].

Traditional Search Engines also do not have Query Suggestions based on the partial query typed by the user. Having Query Suggestions/Query Autocomplete elevates the user's experience as well as provides for easier search of information. We propose to use a Convolutional Latent Semantic Model [2] based system for candidate ranking in Query Autocomplete System. The Query Autocomplete System suggest different queries based on the partial query inputted by the user.

2 Problem Definition

The goal of the project is to design and implement a search engines as a enhancement in various aspects like spell-check, information-retrieval and so on, compared to the search engines build earlier in the assignments and then to provide the results of the search engine under various hypothesis.

3 Motivation

In this huge data driven world right now with lot of information there comes a very important tasks to retrieve these data to use them when required. Using generic search engines would not be the best use case for domain specific data. Thus we require customized search engines based on different domain. The submodule used in the search engine like spell checkers can be used in various other real world problems like grammar correction in documents and so on. As also the project is inspired from real world problems working on this could make us find innovative solutions that can actually create impact.

4 Background and Related Work

In the previous assignments in this course (CS6370: Natural Language Processing), Vector Space Model was used as the Information Retrieval System. Vector space model is an algebraic model for representing text documents as vectors of identifiers. In information retrieval, tf–idf (term frequency–inverse document frequency) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The basic intuition behind tf-idf weighting is that: a word is given more importance if it appears abundantly in a document while the same word loses its importance if it appears more frequently in the corpus (collection of documents). Thus a word with rare appearances in the corpus bears more relevance to a specific document if it appears in that document than some other word which appears a lot in the document while appearing equally or more abundantly in the corpus [3]. Though the vector space model is a robust system, it has many limitations:

– Words in documents are assumed to be independent (especially since we are using only uni-grams). The VSM does not model Word Relatedness and as a result can result in irrelevant documents being retrieved, this reduces precision.
– Documents with similar content but different vocabularies result in small cosine similarity. Since such documents are also relevant but are not retrieved, this reduces recall.
– Computationally intensive: Calculating TF and IDF is the bottleneck of operations and takes a huge amount of time to execute.
– Since the VSM is a uni-gram approach, it doesn't take into account Collations and Co-occurances of words into consideration.

There are also methods such as Latent Semantic Analysis (LSA), WordNet based IR Systems, etc. which do model Word Relatedness similar to Explicit Semantic Analysis (ESA). But as mentioned in the paper [4] (by Evgeniy Gabrilovich and Shaul Markovitch), Wikipedia-based Explicit Semantic Analysis outperforms other methods such as Latent Semantic Analysis, WordNet based IR Systems, etc. on both word/text relatedness (calculated in correlation with humans). This is the reason we plan to use ESA-Wikipedia based IR System in our project.

5 Proposed Methodology

5.1 Spellcheck

Contextual spell checker using BERT, wile learning BERT model looks for the [MASK] tokens and then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. During testing the incorrect words are masked and the other non masked words are used to predict the masked words. In the current project we used pre-trained contextual spell checker. The short coming of this spacy.io model is that it's more focused on the non-word spelling.

Example:

Input Text (with spelling error):
"Income was $9.4 melon compared to the prior year of $2.7 milion."

Output Text (after spellcheck):
"Income was $9.4 million compared to the prior year of $2.7 million."

As seen above can not only detect and fix non-words (milion $\rightarrow$ million) but also word spelling errors (melon $\rightarrow$ million) based on the context of the sentence.

5.2 Information Retrieval - ESA Wikipedia

Explicit Semantic Analysis (ESA) represents meaning in a high-dimensional space of concepts, automatically derived from human-built repositories such as Wikipedia. In ESA Wikipedia, each word is represented by a vector storing the word's association weights to Wikipedia-derived concepts. Each concept is generated from a single Wikipedia article, and is represented as a vector of words that occur in this article weighted by their tf.idf score. Once these concept vectors are generated, an inverted index is created to map back from each word to the concepts it is associated with [1]. The working of ESA Wikipedia is shown below:

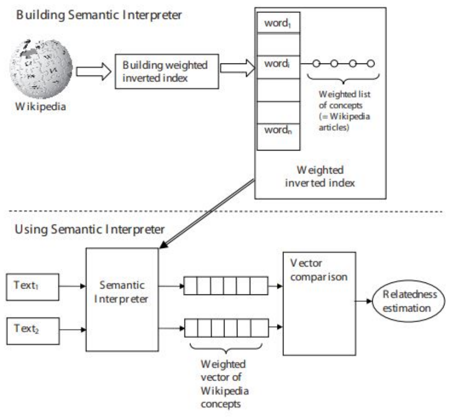
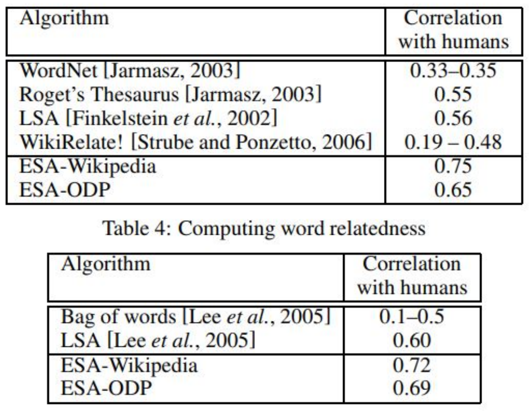

Building Semantic Interpreter
Wikipedia
Building weighted inverted index
word$_1$
word$_i$
word$_n$
Weighted list of concepts (= Wikipedia articles)
Weighted inverted index
Using Semantic Interpreter
Text$_1$
Text$_2$
Semantic Interpreter
Weighted vector of Wikipedia concepts
Vector comparison
Relatedness estimation

Fig. 1. Pictorial Representation of Working of Wikipedia-based ESA

Why ESA Wikipedia was chosen: As per the paper [4] (by Evgeniy Gabrilovich and Shaul Markovitch) which explores the performance of different methods such as LSA, WordNet, etc., ESA Wikipedia performed the best in terms of word/text relatedness. Results are shown below:



Algorithm	Correlation with humans
WordNet [Jarmasz, 2003]	0.33–0.35
Roget's Thesaurus [Jarmasz, 2003]	0.55
LSA [Finkelstein *et al.*, 2002]	0.56
WikiRelate! [Strube and Ponzetto, 2006]	0.19 – 0.48
ESA-Wikipedia	0.75
ESA-ODP	0.65

Table 4: Computing word relatedness

Algorithm	Correlation with humans
Bag of words [Lee *et al.*, 2005]	0.1–0.5
LSA [Lee *et al.*, 2005]	0.60
ESA-Wikipedia	0.72
ESA-ODP	0.69

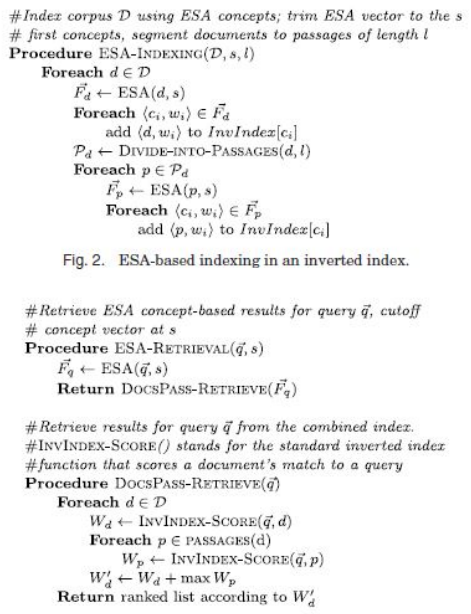
Fig. 2. Comparison of different Semantic Interpreters

Given the described advantages of ESA as a semantic representation and its demonstrated success in other text analysis tasks, it appears well suited for building a successful concept-based IR model. The proposed architecture of the ESA:

Each Document is split into length-based overlapping passages (document concepts). The reasons for doing this:

- A small part of a long document might be relevant to the current query, but the semantics of this part may be underrepresented in the concepts vector (gotten from ESA) for the full document. Previous research using BOW representation has shown that breaking long documents into shorter passages can improve document retrieval [5] [Callan 1994; Liu and Croft 2002].
- Furthermore, it has often been shown that fixed-length passages yield better results than passages based on syntactic or semantic segmentation [Callan 1994; Kaszkiel and Zobel 2001].

Upon receiving a query, our algorithm first converts it to an ESA concept vector. The representation method is identical to the one by which documents are represented at index time. Then the cosine similarity between the document vectors (passages) and the query vectors are calculated and the score of the best performing passage of a particular document is considered the cosine similarity score for that document.



```
#Index corpus $\mathcal{D}$ using ESA concepts; trim ESA vector to the s
# first concepts, segment documents to passages of length l
Procedure ESA-Indexing($\mathcal{D}, s, l$)
    Foreach $d \in \mathcal{D}$
        $\vec{F_d} \leftarrow \mathrm{ESA}(d, s)$
        Foreach $\langle c_i, w_i \rangle \in \vec{F_d}$
            add $\langle d, w_i \rangle$ to $InvIndex[c_i]$
        $\mathcal{P}_d \leftarrow$ Divide-into-Passages($d, l$)
        Foreach $p \in \mathcal{P}_d$
            $\vec{F_p} \leftarrow \mathrm{ESA}(p, s)$
            Foreach $\langle c_i, w_i \rangle \in \vec{F_p}$
                add $\langle p, w_i \rangle$ to $InvIndex[c_i]$
```

Fig. 2. ESA-based indexing in an inverted index.

```
#Retrieve ESA concept-based results for query $\vec{q}$, cutoff
# concept vector at s
Procedure ESA-Retrieval($\vec{q}, s$)
    $\vec{F_q} \leftarrow \mathrm{ESA}(\vec{q}, s)$
    Return DocsPass-Retrieve($\vec{F_q}$)

#Retrieve results for query $\vec{q}$ from the combined index.
#InvIndex-Score() stands for the standard inverted index
#function that scores a document's match to a query
Procedure DocsPass-Retrieve($\vec{q}$)
    Foreach $d \in \mathcal{D}$
        $W_d \leftarrow$ InvIndex-Score($\vec{q}, d$)
        Foreach $p \in$ passages(d)
            $W_p \leftarrow$ InvIndex-Score($\vec{q}, p$)
        $W'_d \leftarrow W_d + \max W_p$
    Return ranked list according to $W'_d$
```

Fig. 3. Algorithmic Representation of Implementation

5.3 Query Autocomplete

There are two parts to Query Autocomplete namely, Candidate Generation and Candidate Ranking. The proposed methods for each of the following are mentioned below.

Candidate Generation: From every query in the search engine logs we generate all possible n-grams from the end of the query. For example, from the query"bank of america" we generate the suffixes "america", "of america"and "bank of america". Next, for a given prefix we extract the end-term. We match all the suffixes that start with the end-term from our precomputed set. These selected suffixes are appended to the prefix to generate synthetic suggestion candidates.

Candidate Ranking: For ranking the generated candidates we use a Convolutional Latent Semantic Model (CLSM) as mentioned in the paper [**?**] (by Yelong Shen et. al). The latent semantic model incorporates a convolutional-pooling structure over word sequences to learn low-dimensional, semantic vector representations for search queries and Web documents.

We adopt the CLSM by training on a prefix-suffix pairs dataset (instead of query-document titles). The training data for the CLSM is generated by sampling queries from the search logs and splitting each query at every possible word boundary. Now given a prefix P and a suggestion candidate C, we extract a normalized prefix p and a normalized suffix s. Then we use the trained CLSM model to project the normalized prefix and the normalized suffix to a common n-dimensional (128-dimensional) space and compute a clsmsim feature. The clsmsim feature used here is cosine similarity between the two n-dimensional vectors. [6]

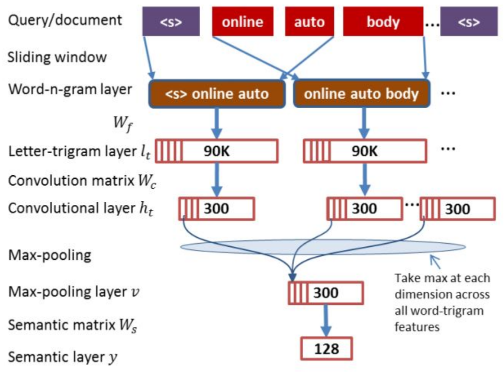


Fig. 4. Architecture of the CLSM

5.4 Evaluation

For Evaluation of our Information Retrieval Model, we use Normalized Discounted Cumulative Gain (nDCG) and for our Query Autocomplete system we use Success Rate Method.

nDCG: The basic intuition behind of nDCG is as such: Very relevant results are more useful than somewhat relevant results which are more useful than irrelevant results. The nDCG algorithm is implemented as follows:

$$DCG_n = \sum_{i=1}^{n} \frac{rel_i}{log_2(i+1)} \tag{1}$$

$$nDCG_n = \frac{DCG_n}{IDCG_n} \tag{2}$$

where,
rel_i is the releveance measure of i^{th} document
$IDCG_n$ is the Ideal DCG at n.

For each query the document ranking is outputted and is matched against the ideal document ranking for that query present under `cran_qrels.json` and the nDCG score is calculated.

Success Rate at k: The Success Rate@k is the average percentage of relevant queries ranked at or above the position K in the ranked list produced by the Query Autocomplete module. In our project, a query is taken as relevant if and only if the QAC system produces an exact match to the input query.

The first N words (N set as 6) of the query is split and inputted into the Query Autocomplete Module. The Query Autocomplete Module outputs a ranked list of query completions upon which succes rate calculations are made.

6 Experiments

6.1 Information Retrieval System

The Information Retrieval System used here is ESA-Wikipedia (as mentioned above).

- In our project the concept space of the Semantic Interpreter was generated by a set of 100 Wikipedia articles (file: *enwiki-20210220-pages-articles-multistream1.xml-p1p41242*).
- The IR System was evaluated on the Cranfield Database (Documents, Queries and Relevance Data) and the Precision@k, Recall@k, MAP@k (Mean Average Precision) and nDCG@k was reported.

6.2 Query Autocomplete System

The Query Autocomplete system here is the CLSM-based system mentioned in the paper (by Yelong Shen et. al)

– The CLSM models are trained using prefix-suffix pairs on the AOL and the Bing query-logs. [6]
– The Query Autocomplete System was evaluated on the queries present in the Cranfield Database based on Success Rate@k metric. That is, the average percentage of relevant queries ranked at or above the position K in the ranked list of QAC.
– The first N words (in our project N=6) of each query is fed into the QAC System and the Success rate@k score is calculated.

7 Results

7.1 Information Retrieval System

The results of the proposed experiment for the IR system evaluated on the Cranfield Dataset is given below. The Semantic Interpreter here is only generated by a concept space spanned by 100 articles.

Table 1. k-value vs Precision@k, Recall@k, F-Score@k, MAP@k, nDCG@k

k-value	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k
1	0.1244	0.0182	0.0310	0.1244	0.1244
2	0.1244	0.0349	0.0523	0.1733	0.1844
3	0.1362	0.0573	0.0767	0.2059	0.2341
4	0.1255	0.0711	0.0857	0.2172	0.2557
5	0.1208	0.0848	0.0940	0.2281	0.2789
6	0.1148	0.0960	0.0989	0.2353	0.2946
7	0.1085	0.1069	0.1017	0.2410	0.3070
8	0.1033	0.1150	0.1029	0.2392	0.3096
9	0.0958	0.1187	0.1002	0.2416	0.3165
10	0.0902	0.1235	0.0985	0.2427	0.3216

As seen above the IR System reaches a nDCG score of 0.32 at k=10. Since the concept space of the Semantic Interpreter is spanned by only 100 articles, the score obtained is low compared to if the concept space was spanned by a larger set of articles (for the whole Wikipedia English Dump, the concept space's dimension would be >10,000). Unfortunately in our project we weren't able to process and create inverted indices for the whole Wikipedia English Dump in the time given so we resorted to 100 articles.

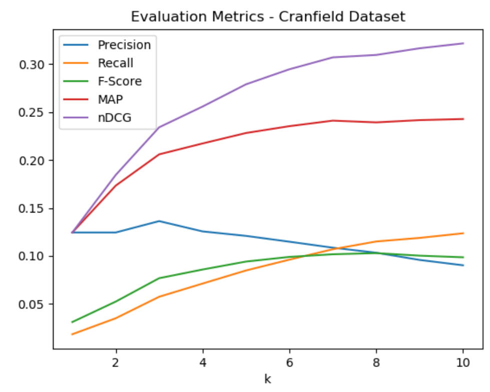


Fig. 5. Evaluation Metrics Plot

Example Query and Retrieved Document:

Query:
"what are the structural and aeroelastic problems associated with flight of high speed aircraft"

Top Document Retrieved:
"the effect of turbulence on slider bearing lubrication . based on prandtl's mixing-length mechanism, the pressure equation for turbulent flow in slider-bearing lubrication is derived . an analytical solution is given and compared with the one for laminar flow . it is found that the turbulent effect increases the pressure and consequently, the load-carrying capacity. however, the power loss also increases"

As seen above the Semantic Interpreter is able to make relations outside just word matching.

7.2 Query Autocomplete System

The results of the proposed experiment for the CLSM-based Query Autocomplete System on queries under Cranfield Dataset is given below. The CLSM here is trained on prefix-suffix pairs on the AOL and the Bing query-logs.

Table 2. k value vs Success Rate at k

k-value	Success_Rate@k
1	0.9216
2	1.38248
3	1.38248

Example Partial Query and Suggestions:

Partial Query:
"structural and aero"

Suggestions:

- "structural and aerodynamic problems"
- "structural and aerodynamic heat transfer to conical bodies for both laminar and turbulent flow"
- "structural and aerodynamic coefficients during re-entry"

8 Conclusion

In this paper we have presented a reliable and effective Text-Based Search Engine whose Information Retrieval System is based on Wikipedia-based Explicit Semantic Analysis (ESA) and Query Autocomplete System is based on a Pre-trained Convolutional Latent Semantic Model which helps give query suggestions. We've also implemented an effective Spell Checker which can not only correct non-words but can do context based spellcheck too.

8.1 Scope for Improvement

Even though the search engine performs really well, it is far from perfect. There are a few areas upon which we can improve on:

- **ESA Wikipedia:** The concept space for the particular project was only built on 100 articles. Wikipedia has over 10,000 articles (in English) from which we can build our concept space. This was not done with the whole of 10,000+ articles since we were short on time. Having a concept space generated by more than 10,000 articles will give much better results than what were shown in this paper.
- **Query Autocomplete:** The Convolutional Latent Semantic Model was trained on AOL and Bing query logs. The queries of the Cranield database are very much different from normal queries received by AOL or Bing. Thus, training the CLSM on the Cranfield Query Database would show much better results than what was shown in the paper.

References

1. OFER EGOZI, SHAUL MARKOVITCH, and EVGENIY GABRILOVICH: Concept-Based Information Retrieval Using Explicit Semantic Analysis; Technion—Israel Institute of Technology
2. Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, Grégoire Mesnil: A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval.

3. Ishwor Timilsina: Vector Space Model (TF-IDF Weighting); https://ishwortimilsina.com/vector-space-model/
4. Evgeniy Gabrilovich and Shaul Markovitch: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis; Israel Institute of Technology
5. Callan 1994; Liu and Croft 2002: Passage retrieval based on language models
6. Bhaskar Mitra and Nick Craswell: Query Auto-Completion for Rare Prefixes