A Toy Search Engine

What are we going to search on?

A Search Engine is an information retrieval system

Typically used to locate information from text or databases

This toy search engine is meant to search on a collection of 5 documents

Documents

Doc1 - please eat an apple

Doc2 - she ate ann egg

Doc3 - he ate all the apples

Doc4 - she likes to siing

Doc5 - Apple was founded in 1976

Tokenize your text

Doc1 - please, eat, an, apple

Doc2 - she, ate, ann, egg

Doc3 - he, ate, all, the, apples

Doc4 - she, likes, to, siing

Doc5 - Apple, was, founded, in, 1976

Some more pre-processing...

Spell check

Suffix stripping

Stopword removal

After spellcheck

Doc1 - please, eat, an, apple

Doc2 - she, ate, **an**, egg

Doc3 - he, ate, all, the, apples

Doc4 - she, likes, to, **sing**

Doc5 - Apple, was, founded, in, 1976

Stemming / Lemmatization

- Stemming: process of reducing inflected (or sometimes derived) words to their word stem, base or root form. Stemming usually refers to a crude heuristic process. There are several types of stemming algorithms.
  - *Suffix-stripping*: Suffix stripping algorithms do not rely on a lookup table that consists of inflected forms and root form relations. Instead, a typically smaller list of "rules" is stored which provides a path to find the root form of a given word.
- Lemmatization: arriving at the base or dictionary form of a word (called lemma) by employing proper morphological analysis and knowledge of vocabulary to remove inflectional endings only.

After Lemmatization

- Doc1 - please, eat, an, apple
- Doc2 - she, **eat**, **an**, egg
- Doc3 - he, **eat**, all, the, **apple**
- Doc4 - she, **like**, to, **sing**
- Doc5 - Apple, was, founded, in, 1976

After stop word removal

Stopwords assumed = {the,a, is,was, an, and, in, on, ...}

- Doc1 - please, eat, ~~an~~, apple
- Doc2 - she, **eat**, ~~an~~, egg
- Doc3 - he, **eat**, all, ~~the~~, **apple**
- Doc4 - she, **like**, ~~to~~, **sing**
- Doc5 - Apple, ~~was~~, founded, ~~in~~, 1976

Stop words are generally the most common words in a language; there is no single universal list of stop words; Any set of words that suits the task in hand can be chosen as the stop words for that given purpose.

Matching

Next, we need to match the query to the documents. In our toy search engine, we are using a simple inverted index for matching.

Please -> Doc1

Eat -> Doc1, Doc2, Doc3

Apple -> Doc1, Doc5, Doc3

She -> Doc2, Doc4

Egg -> Doc2

He -> Doc3

All -> Doc3

Like -> Doc4

Sing -> Doc4

Founded -> Doc5

1976 -> Doc5

Interface to IR System

Documents --> Pre-processing --> Inverted Index

Pre-processing <-- Query

Query

ate an applle” ==> ?

Query

“ate an applle” ==> ?

“ate an applle” ==> ate, an, applle (tokenize)

Query

“ate an applle” ==> ?

“ate an applle” ==> ate, an, applle (tokenize)

“ate an applle” ==> ate, an, apple (spellcheck)

Query

“ate an applle” ==> ?

“ate an applle” ==> ate, an, apple (tokenize)

“ate an applle” ==> ate, an, apple (spellcheck)

“ate an applle”=> eat, an, apple (lemmatization)

Query

“ate an applle” ==> ?

“ate an applle” ==> ate, an, applle (tokenize)

“ate an applle” ==> ate, an, apple (spellcheck)

“ate an applle”=> eat, an, apple (lemmatization)

“ate an apple” ==> eat, apple (stopword removal)

(There is no strict rule as to when tokenization is to be done... can be done before or after pre-processing as it suits the task)

Retrieval from the Inverted Index

- Please -> Doc1
- Eat -> Doc1, Doc2,Doc3
- Apple -> Doc1, Doc5, Doc3
- She -> Doc2, Doc4
- Egg -> Doc2
- He -> Doc3
- All -> Doc3
- Like -> Doc4
- Sing -> Doc4
- Founded-> Doc5
- 1976 -> Doc5

Retrieved Documents

Doc1 - please eat an apple

Doc2 - she ate ann egg

Doc3 - he ate all the apples

Doc5 - Apple was founded in 1976

Ranking

- Doc1 - please eat an apple (score=2)
- Doc3 - he ate all the apples (score =2)
- Doc2 - she ate ann egg (score =1)
- Doc5 - Apple was founded in 1976 (score =1)

Can you guess the scoring mechanism being used here?

How good is our search engine?

- Evaluation measures
  - ?
  - ?
  - ?