This paper serves as a minimal description of some of the algorithms used in the Search Engine project.

Word Stemming:

We need to know first what is suffix stripping which is the operation of removing suffixes from the terms. So, if we considered the document as just set of words {…, connect, …, connecting, …, connected, …, connections, …} after performing the stemming process we should get another set of words like {…, connect, …} we should basically get the root term for each of those words. Also, the process is very independent on the language. In our case we are going to only focus on the English. We used what is known as porter stemmer. This algorithm consists of 5 sets of rules which should be applied in order.

Porter Stemmer Definitions:

1. Consonants: A, E, I, O, U and Y preceded by a constant.
2. Vowels: which are simply any other character.

Following those definitions, we can find that all English words are of the form: $(C)(VC)^m(V)$

1. C = String of one or more consonants.
2. V = String of one or more vowels.
3. M = measure of word or word part when having this form (VC).

Then in order to remove a suffix from we just apply the following rule: *(condition) (S1 – suffix) (S2 – replacement)*, which for example if we have the (m > 1) (EMENT) ( ), that should mean if we have a word with (m > 1) ends with the suffix (EMENT) we should replace this suffix with an empty string as for the word (Replacement), after applying the rule as the condition is fulfilled, we will get (Replac).

Note that the condition part of the previous rule may be something else rather than just be over the (M measurement) and it could be a compound statement rather than an atomic one like the following examples:

1. (*D) – which means the word ends in a double consonant.
2. ((m > 1) and (*s or *t)) – testing the word for having (m > 1) and ending in s or t.

The whole process in order:

1. Step 1:
    a. A- Apply some rules without conditions like:
    (SSES → SS), (IES → I), (SS → SS), (S → ( ))
    b. Apply other set of rules but this time having that condition part like: ((m > 1) (EED → EE)), ((*v*) (ED → ()), ((*v*) (ING → ())
    c. If the rules in the latter step were applied then we can clean up some miss by performing those rules:
    (AT → ATE), (BL → BLE), ((m = 1 & *O) → E), ((*v*) Y → I), ((*d & !(*L or *S or *Z) → Single letter),

2. This Step is called Derivational Morphology: which also about applying some different set of rules like:
    ((m > 0) (ATIONAL → ATE)), ((m > 0) IZATION → IZE),
    ((m > 0) BILITE → BLE)

3. Derivational Morphology phase 2:
    ((m > 0) (ICATE → IC)), ((m > 0) FUL → ( )), ((m > 0) NESS → ( ))

4. Derivational Morphology phase 3:
   ((m > 0) ANCE → ( )), ((m > 0) ENT → ( )), ((m > 0) IVE → ( ))

5. Step 5:
   a. ((m > 0) E → ( )), ((m = 1 & !*o) NESS → ( ))
   b. (m > 1 & *d *L) → Single Letter

Stop Words

This one was a much simpler one. One way to get this feature working correctly is to store all the language stop words in a file and you can check if the word under test is one of those words or not.

Note the previous two algorithms should applied before doing the indexing part as that should reduce the space needed to store the words like in database or something else and will reduce the time needed to search those words later in the web-application.

Robot

It gets the domain of the current URL we will crawl to get it all blocked URLs (the URLs which the domain restricts from being crawled), and add all of this URLs.