Tokenizing and Stemming

Diksha Sharma (dxs134530)
CS 6322.001 – Information Retrieval – fall 2015
Assignment 1 - Report

APPROACH FOLLOWED

- **Stop Words** – The stop words don't carry any information on their own and thus don't contribute anything to the semantics. However, since there could be search queries that could entirely consist of stop words or may have majority of its content as stop words – I have included the stop words as tokens.
- **Numbers** - Since numbers don't contribute to the semantics of the content alone – I am removing words that are entirely numbers from the list of tokens.
- **Words with numbers and characters** – Words which are a combination of only upper case letters and numbers e.g.: flight numbers, drug names etc. I consider them as tokens. These tokens undergo all the special character removal like all other words.
- **Acronyms** – Once the special characters are removed the tokens will be condensed to all character tokens (except for the ones containing numbers along with upper case letters). If the token has all upper case letters it is treated as a new token.
- **Possessives ('s)** - Since all possessives relate to the word original word so I am ignoring all possessives in the tokens but keeping their original words. I am checking for the words ending with 's and removing 's. I am not doing this check for 'S (upper case S) as it might not be a possessive. This will not affect the words that have ' at different location than at the end or end with ' followed by a different character.
- **Special Characters** – Following special characters are removed from the word:

!	%	)	|	[	~
@	^	_	\	]	`
$	&	-	/	:	?
#	*	+	{	;	.
	(	=	}	'	,

- **Contractions** – Contractions like you're etc. will be converted to all character tokens after removing all special characters.
- **Case** – Once all checks to remove unwanted tokens are performed – all tokens are converted to lower case except tokens with all upper case letters so USA, U.S.A. are same and u.s.a. and usa are considered same tokens.
- **Null Strings** – Since these are just white space – I ignore them.
- **Strings of length = 1** – Since these are just one character strings and have no semantic meaning – I ignore them with exception to "a" as it is a stop word and I am considering stop words as tokens.
- **Spaces** – I trim the word of any spaces before I start any checks for tokens or other characters. All spaces are ignored.
- **SGML Tags** – All words that start with < and/or end with > are considered SGML tags and are ignored.
- **SGML Tag Attributes** – Since the collection given does not have any tag attributes – my code submission does not exclude them from the list of tokens. They will be handled as any other word in the documents.

MAJOR ALGORITHMS AND DATA STRUCTURES USED

I have used Array lists to create token structure which stores the token and its frequency in the collection. I have used array lists so I can dynamically add new tokens as required in the list instead of predefining the array since I am unaware of its size.

Steps followed for tokenizing:

1. Read file and repeat the following process for all words extracted from the file:
    a. Trim to remove spaces if any.
    b. Check if the word has any numbers.
        i. If it does not have any numbers then:

1. Check if it is a tag word – starting with < and/or ending with > - ignore such words
2. If not a tag word then check if it is a possessive (containing 's). If yes remove 's and keep the remaining word.
3. Check if the word has any special characters – if yes then remove the special character and check for remaining special characters.
4. Once all special characters are removed – ignore any null strings.
5. For words which has characters – if their length is equal to 1 – ignore then with exception to article "a".
6. Once a word passes all above test check if it has already been added to the tokens list. If yes then increase its frequency by 1 else add it to the list as a new token with frequency = 1.

ii. If the word has any numbers then:

1. Remove all numbers from the word.
2. Check if the characters are all upper case. If yes then:
   a. Ignore any tag words starting with < and/or ending with >.
   b. If not a tag word check if it ends with 'S – Most likely since all characters are upper case so 'S is not a possessive – so we keep the S at the end.
   c. If the word does not end in 'S then next I check for all special characters are remove them.
   d. Ignore any null strings
   e. If not a null string check if it equals "A" – most likely it is an article – convert to lower case.
   f. If length of the word is greater than 1 – check if it is already added as a token in the list – If not add it as a new token with frequency = 1 else increase the token frequency by 1 for already added token.
3. If the characters are a mix of upper case and lower case letters or all lower case letters then:
   a. Ignore any tag words
   b. If not a tag word then remove any possessive 's from its end.
   c. Remove all special characters.
   d. Ignore any empty string.
   e. If length is greater than 1 then check if already added as a token.
   f. If length is equal to 1 and it is "a" then increase frequency of article "a".

c. I am ignoring all strings of length = 1 and content not equal to article "a".

2. Run porter algorithm on the same files and fetch the stems based on porter algorithm.

ANSWERS TO QUESTIONS IN ASSIGNMENT

Following are the questions asked in assignment and their answers based on my implementation:

The number of tokens in the Cranfield text collection

222864 Tokens (Including stop words)

The number of unique (e.g. distinct) tokens in the Cranfield text collection

10154 Tokens (Including stop words)

The number of tokens that occur only once in the Cranfield text collection

4562 Tokens (Including stop words)

The 30 most frequent word tokens in the Cranfield text collection – list them and their respective frequency information

Token: the Frequency: 19442
Token: of Frequency: 12672
Token: and Frequency: 6660
Token: a Frequency: 5933
Token: in Frequency: 4642
Token: to Frequency: 4529
Token: is Frequency: 4111
Token: for Frequency: 3490
Token: are Frequency: 2427
Token: with Frequency: 2263
Token: on Frequency: 1940
Token: at Frequency: 1834
Token: by Frequency: 1748
Token: flow Frequency: 1736
Token: that Frequency: 1565
Token: an Frequency: 1386
Token: be Frequency: 1271
Token: pressure Frequency: 1133
Token: from Frequency: 1116
Token: as Frequency: 1111
Token: this Frequency: 1080
Token: which Frequency: 974
Token: number Frequency: 964
Token: boundary Frequency: 897
Token: results Frequency: 885
Token: it Frequency: 852
Token: mach Frequency: 817
Token: theory Frequency: 775
Token: layer Frequency: 728

The average number of word tokens per document

159 tokens per document (Including stop words)

How long the program took to acquire the text characteristics

25 sec (Tokenizing) average

1 sec (Stemming)

The number of distinct stems in the Cranfield text collection

4543 (206038 total stems)

The number of stems that occur only once in the Cranfield text collection

1527

The 30 most frequent stems in the Cranfield text collection – list them and their respective frequency information

Token: the Frequency: 18695
Token: of Frequency: 11356
Token: and Frequency: 5711
Token: a Frequency: 5333
Token: to Frequency: 4404
Token: in Frequency: 4254

Token: is Frequency: 4110
Token: for Frequency: 3268
Token: ar Frequency: 2428
Token: with Frequency: 2084
Token: on Frequency: 1865
Token: flow Frequency: 1705
Token: by Frequency: 1703
Token: at Frequency: 1592
Token: that Frequency: 1570
Token: be Frequency: 1366
Token: an Frequency: 1261
Token: pressur Frequency: 1251
Token: number Frequency: 1246
Token: as Frequency: 1102
Token: thi Frequency: 1080
Token: result Frequency: 1073
Token: from Frequency: 1067
Token: it Frequency: 1034
Token: boundari Frequency: 998
Token: which Frequency: 969
Token: layer Frequency: 948
Token: effect Frequency: 849
Token: method Frequency: 824
Token: theori Frequency: 793

The average number of word stems per document

78

How the program handles:

A. Upper and lower case words (e.g. "People", "people", "Apple", "apple")

After all special characters are removed – I reduce the words to lower case (unless they are in all upper case). So People and people will be same and Apple and apple will be treated as one token.

B. Words with dashes (e.g. "1996-97", "middle-class", "30-year", "teen-ager")

The program removes all dashes from the words.
"1996-97" – Is all numbers – so will be ignored
middle-class" – Will be converted to "middleclass"
"30-year" – The dashes will be removed, numbers will be removed and the remaining token will be processed. So it will be reduced to "year".
"teen-ager" – Will be converted to "teenager"

C. Possessives (e.g. "sheriff's", "university's")

Removes all 's from possessives.
"sheriff's" – will be stored as "sheriff"
"university's" – will be stored as "university"

D. Acronyms (e.g., "U.S.", "U.N.")

After removing all special characters all words containing only upper case letters with/without numbers are stored as they are and not converted to lower case.
"U.S." – will be stored as "US"
"U.N." – will be stored as "UN"