Analysis of Stemming Algorithms – Comparing Modern Day Stemmers' Performance in IR Systems

Nisha Choudhary, Maria Vivanco, and Saumyaa Mehra

CS380 – Recent Advances in Computer Science: Information Retrieval & Web Search

1. Introduction

In information retrieval systems, two of the biggest issues faced are storage and accuracy. Stemmers attempt to solve both of these issues. By stemming queries and documents in IR systems, IR systems can rank and retrieve documents more accurately. For example, if someone is searching for “Cars for Sale”, documents containing “cars” and the singular form “car” should be retrieved.

This report aims to answer the question of whether light or heavy stemmers are better. We answer this question by testing the performance of three popular stemmers in query based information retrieval systems. The three stemmers we test are: Porter, Snowball, and Krovetz. We are interested in how document ranking differs pre and post stemming. Does stemming indeed help with retrieval accuracy? How does the aggressiveness of the stemmer help/hinder retrieval accuracy?

2. Background

In this section, we will describe the three different stemming algorithms that we chose: Porter, Snowball, and Krovetz.

2.1 Porter Stemmer

One of the stemmers we are analysing is the Porter Stemmer. This stemmer was created by Martin Porter in the 1980s and we chose to work with it because it is the most widely used stemmer[1]. This stemmer takes a rule based approach to stemming and usually returns non-real words. The algorithm of the porter stemmer uses a series of 5 steps to stem words.

Each of these steps takes into account the word-ending, and based on a set of conditions – if the word ending fulfils the condition, the ending is either kept, removed, or replaced with something else –

- (condition) S1 → S2

This step-by-step stripping is based on the following conditions:

- *S – the stem ends with S (and similarly *letter for other letters)
- *v* – the stem contains a vowel
- *d – the stem end with a double consonant (e.g. -TT, -SS)
- *o – the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP)
- m = measure of word = # of occurrences of vowels (V) and consonants (C) next to each other, in order VC (e.g. row = r(ow)- m=1)

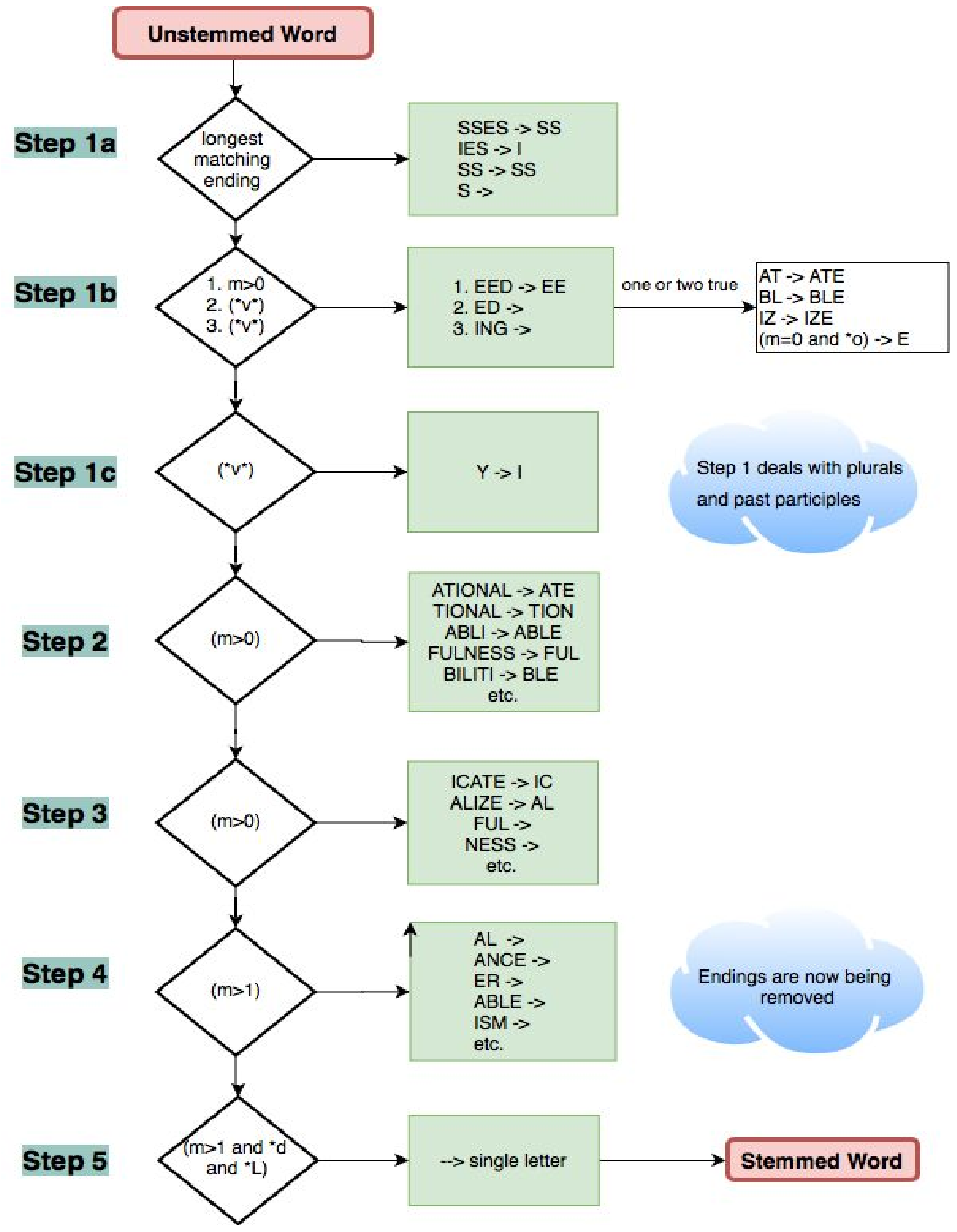

Diagram 1: Layout of Porter Stemmer Algorithm

2.2 Snowball Stemmer

The second stemmer we analyze is the Snowball Stemmer, developed by Martin Porter, an enhanced version of the Porter Stemmer and is commonly referred to as Porter 2. The Snowball Stemmer is built upon the Porter Stemmer with additional rules. This stemmer is cited to have a 5% difference from Porter. After comparing our stemmed datasets by each stemmer, we realised that the additional changes that the Snowball stemmer implemented are:

1. changes terminating y’s less often
2. suffixes like -us are not losing their -s as often as it would in the regular Porter stemmer
3. correctly removes additional suffixes (e.g. the -ly suffix),
4. deals with apostrophes by removing them and treating them as a suffix

These four differences and enhancements allow the Snowball Stemmer to be slightly more accurate than the original Porter Stemmer.[citation] The Snowball stemmer also does not produce jargon words i.e. it returns proper dictionary words or stems more often than the Porter stemmer.

Due to the aforementioned reasons, we chose the Snowball Stemmer because we believe that it could be more effective than the Porter Stemmer. We wanted to further test and verify this theory and see how it would perform in document retrieval and text compression.

2.3 Krovetz Stemmer

The third stemmer that we are considering is the Krovetz Stemmer developed in 1993 by Robert Krovetz. Unlike the last two stemmers we discussed, the Krovetz stemmer uses a dictionary based approach to stemming. It uses a built in English language dictionary to attempt to reach the root word. It is often referred to as a lemmatizer because of its light stemming i.e. post-stemming, it returns proper dictionary words instead of just non-dictionary stems of words.

The Krovetz stemmer checks the word against the dictionary and proceeds stemming if not found. If the word is present, it is either left alone or checked against possible suffixes. If the word is not found, it checks for suffixes that could be removed. The removed word is then checked against the dictionary. Unlike the last two stemmers, this stemmer produces words rather than stems.

We chose the Krovetz stemmer because it took a completely different approach as a partial lemmatizer as compared to the other two stemmers, and is also known as a light stemmer. This would help us compare light stemmers with heavy stemmers and see how they perform in document retrieval and text compression.

3. Algorithm

In this section, we present our algorithms for our stemming aggressiveness test as well as our IR system.

3.1 Stemming Aggressiveness

In order to conclude whether light or heavy stemmers are better, we must first test the aggressiveness of each of our stemmers. We measured this by stemming each document in the *New York Times* corpus mentioned in Section 6.3 and seeing how the number of unique tokens change after stemming.

These algorithms are how we stem each corpora using each stemmer:

```
Function porter_stemmer(input_file, output_file) {
      Open input_file
      For each line input file:
           Strip punctuation and normalize to lower case
           porter(normalized line)
           Write line to output_file
}
```

Snowball:

```
Function snowball_stemmer(input_file, output_file) {
     Open input_file
     For each line input file:
          Strip punctuation and normalize to lower case
          snowball(normalized line)
          Write line to output_file
}
```

Krovetz

```
Function krovetz_stemmer(input_file,output_file) {
     Open input_file
     For each line in input file:
          String punctuation and normalize to lower case
          Krovetz (normalized line)
          Write line to output_file
}
```

Once we have the stemmed version of the corpora, we pass it into our second algorithm getStats where we output how many unique tokens are in a given file.

```
Function getStats (file) {
     Open file
     Instantiate hashset of tokens
     For each line in file:
            For each word in line
                 If (hashset does not contain word)
                      Add word to hashset
       Return size of hashset
}
```

3.2 Query Based IR System Test

The IR engine starts by reading in each file and standardizing each token (i.e. making sure that every letter is lowercase and trailing punctuation is removed). For each file, it then does the following:

```
(1)  It sorts each unique token by frequency within the document.
(2)  For each unique token in the document
     (a)  Finds all the documents in the corpus that contain the
          token
     (b)  Stores them
     (c)  Calculates the TF-IDF weight
(3)  Finds the results for each query
     (a)  Calculate the cosine similarity of each document for each
          query
          (i)  Relevant documents for a query have a cosine
               similarity that is greater than 0
     (b)  Return each query’s results
```

4. Hypothesis

Given our background research on the stemmers detailed in Section 2, we hypothesize that the most aggressive stemmer would be Snowball because it is built upon Porter and deals with corner cases that Porter doesn’t consider – it considers -ly as a suffix and also does not overstem the ending -s.

We hypothesize because of this aggressiveness, Snowball will have a higher rate of stemming accuracy. Due to this stemming accuracy, we hypothesize that Snowball will produce the most accurate document ranking. Due to this, we hypothesize that Snowball will perform the best in the information retrieval test and thus is the most effective stemmer.

5. Methodology

5.1 Description of Sample Corpora

We selected four sets of documents for the stemming aggressiveness test. By seeing how the stemmers stem these sets of documents, we can see how the stemmers perform in different cases. Our four sets of documents are: Unix English Language Dictionary, New York Times Opinion Articles, Horror Books, and Astrology Books. The statistics of these corpora pre-processing are found in Table 1 (below).

The first corpora we used was the Unix English Language Dictionary. This had 30K most commonly used english words. This is the most optimal use case for the stemmers since they are stemming the English Language in a cleaner manner as compared to stemming our datasets which are very varied. By stemming this dictionary, we can test the actual aggressiveness of each stemmer since it does not have any unexpected/unique words that the stemmers won't expect, and thereby get consistent data and test it against our hypothesis.

Secondly, aside from this, we wanted to see how the stemmers performed in cases where there were unique/unexpected words or on data sets that were very different in nature. To do this, we chose three different sets of corpora that provided unique use cases for the stemmers. The NYT Opinion Articles were written recently and the majority of the words are common words in the English language. However, these articles still have unique words and proper nouns. Though this is not as optimal as stemming the English language, this represents the most common use case for stemmers. In IR systems like Google, they have to deal with proper nouns as tokens. For example, if someone searches "Donald Trump", they would not only want the documents with the exact string "Donald Trump" but also documents with the possessive token "Donald Trump's" and "Trump's". This is a case that stemmers need to account for and that we will be testing by using the NYT corpus.

The Horror books were written in the 1900s and use older english and more unique tokens. We expect that this will provide a more difficult use case for the stemmers since it uses outdated English. The Astrology corpus is made up of books of various branches of astrology and consists of many unique tokens, out of which many are not recognizable words in the English dictionary. Consequently, we expect that this will be the most difficult use case for the stemmers and may fetch us some odd results. By testing on these different cases of corpora, we can see how the aggressiveness of the stemmers changes in various scenarios.

Corpora	**Description of Scope**	**Total # of Documents in collection**	**Total # of Words**	**# of Unique Tokens**
English Dictionary	Unix most common english words	1	30,000	29416
New York Times Opinion Articles	Articles from NYT from September 2019 through February 2020	83	100,315	13287
Horror Books	Books written by authors in the 1900s	94	5,267,539	92894
Astrology Books	Books about multiple forms of astrology (Hindu, etc)	92	172,870	17787

Table 1 – Test Corpora to test Stemming Aggressiveness vector

5.2 Stemmers Implementation

To use the three stemming algorithms discussed under *section 2,* we used imported stemmers. For the Porter Stemmer, we used Python's NLTK built in Porter Stemmer. We passed in the input file into our porter stemmer python script and the script outputted a stemmed file. We followed the same procedure for the other two stemmers. We used the built in Snowball Stemmer function also in the Python NLTK library. For the Krovetz Stemmer, we downloaded the Krovetz Git repo to obtain the stemmer.

5.3 Measuring Aggressiveness

Once we had stemmed each corpora, we measured the number of unique tokens before stemming and after stemming using the getStats algorithm outlined in *section 3.1.* We defined the aggressiveness of a given stemmer (s) on a given data set (d) to be A_{sd} as follows

$$A_{sd} = (U_d - U_{sd}) / (U_d)$$

This equation finds the percentage of text compression that the stemmer did. To find the overall aggressiveness of the stemmers, we took the average over all of the datasets.

5.4 Measuring Stemming Accuracy

In this test, we were interested in how each stemmer was able to group similar words into the same stem. For example, the word 'danger' has several forms: dangerous, dangers, dangered, endangered. We are interested in how many of these words are grouped into the same stem. This was done manually by looking through the three stemmed versions of the english dictionary side by side in Excel.

5.5. Query Based IR System Test

To look at the information retrieval accuracy, we built our IR system using the algorithm described in the Algorithm section (under Query-Based IR System Test) and focused on the opinion articles from *The New York Times*. We decided to use this corpera because it provided the most average and used case for IR systems - a dataset made up of ordinary english as well has plenty of proper nouns and unique/unexpected tokens.

Below are the queries that were fed into our IR system. We chose to feed in the different forms of the word "acknowledge" to see if the IR system would return the same documents for each one. We also chose queries from specific documents to see if the IR system would return the correct document.

List of Original Queries

according to universe
acknowledge accordingly
harvard university
acknowledge their validity
acknowledge acknowledged acknowledgement acknowledges acknowledging acknowledgment acknowledgments
acknowledge
acknowledged
acknowledgement
acknowledges
acknowledging
acknowledgement
acknowledgements
apart from excessive sleepiness after sleeping for so long
accord accordance accorded according accordingly
accord
accordance
accorded
according
accordingly
ordinary civility

To find the results for each stemming algorithm, each of these queries were also stemmed according to its corresponding algorithm when looking at how effective each algorithm is in an IR engine. We manually examined the output of the document ranking from the IR system to see how documents changed ranking/similarity.

6. Testing Results & Analysis

6.1 Stemming Aggressiveness

These are the results after stemming the corpora with each respective stemmer. The percentages represent how much the number of unique tokens decreased after stemming, i.e. the aggressiveness vector A. The most aggressive stemmer was the Snowball stemmer and the lightest stemmer was the Krovatz stemmer.

Data set	**# of unique tokens**	A_{porter}	$A_{Snowball}$	$A_{Krovatz}$
Horror Books	92894	23.70%	24.46%	20.19%
NYT Articles	13287	29.46%	30.24%	21.50%
Astrology Books	17787	27.24%	28.00%	27.95%
English Words	29416	40.86%	42.41%	33.63%
Average	–	30.32%	31.28%	25.82%

Table 2 – The total number of unique tokens and compression rate for each stemmer (per corpus)

<u>*Discussion*</u>

The most aggressive stemmer was the Snowball stemmer and the lightest stemmer was the Krovetz stemmer. Since the Snowball stemmer is built on top of Porter, this stemmer was the most aggressive. Since the Kroverz stemmer takes a dictionary based approach, it only stems until it finds a root in the dictionary. This resulted in a lighter stemmer.

As mentioned before, the horror book data set provided a unique test case: a corpera with old english and many unique tokens. The text compression was lower than the other sets because of its unique vocabulary. The snowball stemmer was the most aggressive out of the three stemmers but not by much (only 1-3% more). The New York Times opinion articles provided a semi optimal use case because it was made up of common english words but still had many unique tokens such as proper nouns and slang. As mentioned, this is the average use case since many documents online have the same landscape of vocabulary. The most aggressive stemmer was once again the Snowball stemmer. Interestingly, it was 8.5 more aggressive than Krovetz. The most difficult use case for the stemmers was the astrology books corpus.. The stemmers performed the same in this case. This is because this corpus's vocabulary had many unique tokens and often non-English tokens.

The most interesting case is how the stemmers stemmed the english dictionary. This case lets us see how much of the english language can be condensed into stems. The stemmers decreased the number of tokens by roughly 40%, which is almost half!

Once we averaged the aggressiveness values for each dataset, we see that the Snowball stemmer is the most aggressive stemmer and the Krovetz is the lightest stemmer. This is in alignment with our hypothesis. Now that we have assigned our "light" and "aggressive" stemmer, we can infer conclusions when comparing how these stemmers perform in the next two tests.

6.2 Stemming Accuracy

In this test, we wanted to go under the hood and see *how* each stemmer is stemming a group of words. These are some interesting cases we found:

Original Word	Porter	Snowball	Krovetz	Preferred Stemmer
acknowledge	acknowledg	acknowledg	acknowledge,	**Porter/ Snowball**
acknowledged			acknowledgement	
acknowledgement				
acknowledges				
acknowledging				
acknowledgment				
acknowledgments				
universal	univers	univers	universal	**Krovetz**
universally			universally	
universe			universe	
university			university	
accord			accord	**Snowball**
accordance			accordance	
accorded			accord	
according	accord		according	
accordingly	accordingli	accord	accordingly	
abudantly	abundantli	abund	abudantly	**Krovetz**

Table 3 – Word groups and how they are stemmed by each stemmer

Discussion

1. Acknowledg* : According to our understanding, we prefer Porter/Snowball over Krovetz because of more text compression in Porter and Snowball i.e. fewer number of stemmed words for a group of very similar words, while keeping the accuracy of the stem intact.
2. Univers*: Even though all these words mean very different things, Porter and Snowball stem all these words to the same stem i.e. 'univers', thereby compromising on accuracy. Meanwhile, Krovetz keeps some words intact and stems the rest retaining the accuracy of the stemmed results, even though we get lesser text compression. Hence, Krovetz is preferable for more accurate results.
3. Accord*: For this example, Snowball deals with the edge case of '-li' ending and stems stem the word to its actual stem. Meanwhile, Krovetz keeps the word in its original form, compromising on text compression. So, Snowball is preferable.
4. Abund*: Snowball again deals with the '-li' edge case better than Krovetz, but here retaining the actual word itself may improve accuracy, and hence Krovetz seems to be better than the rest.

Overall analysis – Which stemmer is preferable also depends on our data set and the kind of words that we're dealing with. Different stemmers could work in different ways on different sets of data, in terms of text compression and stemming accuracy. Here, **Snowball seems to be the best stemmer** in terms of text compression and stemming accuracy.

6.3 Query Based IR System Results

For the queries, we focused on looking at the opinion articles in the *New York Times*. The query results (below) are formatted as follows:

query
results (cosine similarity)

* Note that all of the queries with an asterisk next to it are queries that had more results than what was listed below, and results for all queries are not shown

1. Pre-Stemming
   harvard university *
   Trump Wants Law and Order Front and Center 0.9898824527566188
   Four Key Things You Should Know About Health Care 0.9898824527566187
   Of All Trump's Defenses, This Is the Lamest 0.9898824527566186
   Did I Just Get Yanged? 0.910232324498422
   How Much of Harvard's $40 Billion Endowment Is Invested in Fossil Fuels? 0.8894358020106071
   Why Trump Persists 0.8437736313909587

   Porter
   harvard univers *
   did I just get yanged? 0.9988822529756158 **→ Ranking and score increases**
   trump want law and order front and center 0.9954059143710997
   four key thing you should know about health care 0.9712739692103212
   Of all trump' defenses, thi Is the lamest 0.971273969210321
   whi trump persist 0.869214547476548
   how much of harvard' $40 billion endow Is in lo vest in fossil fuels? 0.862236354767143

   Snowball
   harvard univers *
   did I just get yanged? 0.9988822529756158 **→ Ranking and score increases**
   trump want law and order front and center 0.9954059143710997
   four key thing you should know about health care 0.9712739692103212
   Of all trump' defenses, thi Is the lamest 0.971273969210321
   whi trump persist 0.869214547476548
   how much of harvard' $40 billion endow Is invest in fossil fuels? 0.862236354767143

   Krovetz
   harvard university *
   trump wants law and order front and center 0.9843960045703491
   four key things you should know about health care 0.984396004570349
   of all trump's defenses, this is the lamest 0.984396004570349
   did i just get yanged? 0.9003632756592947

   We chose the query harvard university because university, universe, and universes all stem to univers in the Porter and the Snowball stemmers so that we can look at how the results would compare. Our interesting findings are noted in the observations.

   Observations 1

   - Cosine similarity score for the documents retrieved after stemming with Porter and Snowball is greater than the score pre-stemming

- Ranking of the documents changes after stemming for all the stemmers as compared to original
- Porter and Snowball function identically in our IR system.

2. Pre-Stemming

accord accordance accorded according accordingly *
Democrats' Baffling 2020 Mess 0.6150016733379723
Why Donald Trump Hates Your Dog 0.6150016733379723
We Can't Afford Trump as Our Commander in Chief 0.447213595499958
Why Trump Persists 0.4472135954999579
Coronavirus Spreads, and the World Pays for China's Dictatorship 0.4472135954999579

Porter

accord accord accord accord accordingli *
whi trump persist 0.9922778767136677 → **Ranking and score increases**
the white hous motto: watch your back 0.9922778767136677
bewar the pandem panic 0.9922778767136677
whi donald trump hate your dog 0.9922778767136677
coronaviru spreads, and the world pay for china' dictatorship 0.9922778767136677

Snowball

accord accord accord accord accordingli *
whi trump persist 0.9922778767136677 → **Ranking and score increases**
the white hous motto: watch your back 0.9922778767136677
bewar the pandem panic 0.9922778767136677
whi donald trump hate your dog 0.9922778767136677
coronaviru spreads, and the world pay for china' dictatorship 0.9922778767136677

Krovetz

accord accordance accord according accordingly *
democrats' baffling 2020 mess 0.4146340435791247
why donald trump hate your dog 0.4146340435791247
the white house motto: watch your back 0.3015113445777637
we can't afford trump as our commander in chief 0.3015113445777637
why trump persist 0.3015113445777637

Because this group of word forms has been stemmed slightly differently by the Krovetz stemmer, we wanted to take a look at how this would compare if we looked at the word group. Our interesting findings are noted in the observations.

Observations 2

- Cosine similarity scores for the documents retrieved after stemming with Porter and Snowball are much greater than the score pre-stemming and the score post stemming with Krovetz. Thus, Porter and Snowball stemmers are increasing the accuracy of IR systems.
- Again, ranking of the documents changes after stemming for all the stemmers as compared to original
- Porter and Snowball function identically in our IR system.

3. Pre-stemming

according to universe *

Trump's Digital Advantage Is Freaking Out Democratic Strategists 0.8011326633004587
The Next Decade Will Be Just as Bad 0.7310310741702064
Warren and Klobuchar Teach the Boys a Lesson 0.5773502691896258
What Does It Mean to Have a Serious Drinking Problem? 0.5773502691896258
Did I Just Get Yanged? 0.5773502691896258

Porter

accord to univers *

the sidney award 0.8095169351771415
starr chamber: the sequel 0.8095169351771414
coronaviru spreads, and the world pay for china' dictatorship 0.8095169351771414
pete ain't It 0.8095169351771414
the next decad will Be just as bad 0.8095169351771414

Snowball

accord to univers *

the sidney award 0.8095169351771415
starr chamber: the sequel 0.8095169351771414
coronaviru spreads, and the world pay for china' dictatorship 0.8095169351771414
pete ain't It 0.8095169351771414
the next decad will Be just as bad 0.8095169351771414

Krovetz

according to universe *

trump's digital advantage is freak out democratic strategist 0.8011326633004587
the next decade will be just as bad 0.7310310741702064
china didn't want us to know. now its own file are do the talking. 0.5773502691896258
china's new civil religion 0.5773502691896258
warren and klobuchar teach the boys a lesson 0.5773502691896258

Since the word group for according and the word group for universe are stemmed differently by the Krovetz stemmer, we wanted to look at the original "according to universe" query and how each stemmer would influence which results came up and/or what the cosine similarity would be. Our interesting findings were noted in the observations.

Observations 3

- Cosine similarity scores for the documents retrieved after stemming with Porter and Snowball are much greater than the score pre-stemming and the score post stemming with Krovetz. Thus, Porter and Snowball stemmers are increasing the accuracy of IR systems.
- Porter and Snowball function identically in our IR system.

4. Pre-Stemming

   acknowledge their validity *
   Devin Nunes Is Danielle Steel 0.6482731536635694
   The Future of American Politics 0.6071375154698476
   Mayor Pete's Gay Reckoning 0.5924625643048007
   The American Health Care Industry Is Killing People 0.5773502691896258
   Did I Just Get Yanged? 0.5773502691896258

   Porter
   acknowledg their valid *
   the flaw human of silicon valley 0.6904083374800494
   'bojack horseman' end with a necessari reckon 0.6742218541710693
   the white hous motto: watch your back 0.6385205984959307
   the shaki futur of .org domain 0.6385205984959306
   devin nune Is daniel steel 0.6289295076615469

   Snowball
   acknowledg their valid *
   the flaw human of silicon valley 0.6904083374800494
   'bojack horseman' end with a necessari reckon 0.6742218541710693
   the white hous motto: watch your back 0.6385205984959307
   the shaki futur of .org domain 0.6385205984959306
   devin nune Is daniel steel 0.6289295076615469

   Krovetz
   acknowledge their valid *
   the flaw humanity of silicon valley 0.685288250490692
   the white house motto: watch your back 0.6354414336083095
   the shaky future of .org domain 0.6354414336083095
   devin nun is danielle steel 0.6262963130238783
   the future of america politics 0.6169264603742076

   For similar reasons as the third set of queries, we wanted to look at the query "acknowledge their validity" in order to see how the different stemmed versions of the query lead to a different set of results and/or different cosine similarity measures. Our interesting findings are noted in the observations.

Observations 4

- Cosine similarity scores for the documents retrieved after stemming with Porter and Snowball are much greater than the score pre-stemming and the score post stemming with Krovetz. Thus, Porter and Snowball stemmers are increasing the accuracy of IR systems.
- Porter and Snowball function identically in our IR system.

7. Conclusions

According to our results in section 6, our hypothesis stands true and snowball seems to be the best stemmer out of Porter, Snowball and Krovetz.

1. In terms of stemming aggressiveness, Snowball does a decent job at text compression while also retaining the meaning of the words and not producing non-recognizable stems. Meanwhile, Porter is not as effective as Snowball at text compression, and Krovetz is a much lighter stemmer than both Porter and Snowball.
2. In terms of stemming accuracy (as explained in section 6.2), Snowball again proves to give us the most accurate and clean stems, that also deals with many of the edge cases while Porter does not.
3. For query based IR systems, Porter and Snowball function in an almost identical manner for our datasets. This is not surprising as Snowball is an extension of Porter. Yet, document retrieval after stemming with Porter and Snowball returns documents with higher similarity scores than documents retrieved pre-stemming.

Even though Porter and Snowball function in very similar ways and are aggressive stemmers in nature, they are great at text compression and at improving IR system accuracy. For the aforementioned reasons, in our opinion, Snowball is still better than Porter and the most preferable stemmer out of all three in terms of text compression, stemming accuracy and document retrieval.

References

https://academicpublishingplatforms.com/downloads/pdfs/gnujet/volume1/201107241724_10-46-1-PB.pdf

https://github.com/rmit-ir/KrovetzStemmer

https://tartarus.org/martin/PorterStemmer

https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8