# SI 650 / EECS 549: Homework 2

Uniquename: sijuntao

---

## Task 1.4

For different stemming configuration or stopwords:

- The number of terms after processing is different. The configuration of "PorterStemmer" have smaller number of terms than that of "Stopwords". This may be because besides deleting stopwords, "PorterStemmer" also transforms a word into its root form. Therefore, the number of terms will be less.
- The number of postings and tokens are different. The configuration of "PorterStemmer" have larger number of postings and tokens. This may be because "PorterStemmer" remains the changing form of some stopwords so the number of tokens will be larger.

For different tokenisation:

- The number of terms, postings, and tokens are very close. The configuration of "UTFTokeniser" has almost the same numbers with that of "UTFTwitterTokeniser". The reason may be that they both distinguish UTF words, but twitter has some specific phrases so there exists a little bit difference.
- The indexing time has a little bit difference. "UTFTwitterTokeniser" has larger indexing time.

I'm surprised by the difference of results of using different stemming configuration. There is a large gap between the number of tokens and postings. Using approprite stemming to preprocess the data is very important.

## Task 2.2

From the results, we can find:

- BM25 is the  best model among these five models. It has the highest score in all five metrics.
- PL2 and DPH perform better than Hiemstra_LM and DFIC.

It is possible to achive a better performance with some hyperparameter tuning. The characteristics of different dataset are different, so the hyperparameter tuning may help.

## Task 3.5

From the comparison of BM25 and Pivoted Normalization, we can find:

- BM25 outperforms Pivoted Normalization in all the indexes I created before on the metrics of "map" and "ndcg".

From the performance of BM25 with different hyperparameters, we can find:

- For b, the overall trend is that the larger the value of b is, the greater the ndcg is, that is, the better BM25 performs.

- For k1, the trend is unclear. In the range of 1.2-2.0, the value of ndcg fluctuates a lot with the increasing of k1. The relationship between k1 and the performance is not linear. Therefore, to get best performance, k1 needs to be tuned by experiments.
- For k3, the performance of BM25 does not change a lot when k3 increases from 4 to 12. For this dataset, it seems that the smaller the k3 is ( smaller than 2), the better the overall performance is.

Actually, hyperparameter b is used to adjust the effect of document length on correlation. The larger b is, the greater is the impact of document length on the relevance score, and the smaller is vice versa. There is no strong linear relationship between the performance of BM25 and b, so tuning the value of b is necessary for finding the best model. In addition, the relationship between k1 and k3 are also unclear, so tuning by experiments may help the performance of the model to be better.

## Task 4.2

My function mainly have four parts:

- **IDF**. The IDF I designed is *log2(e) * ln((N+1)/(df+0.5))*. I did not choose the IDF form in BM25 because when df is very large, the value of IDF will become negative, which is problematic. In addition, when df is very small, the IDF form should still be valid, so I add 0.5 to the denominator. The value of 0.5 is a result of tuning among the range of [0,1]. Since the occurence of term in the whole collection is relatively important, I multiple the term with log2(e) to make the influence of IDF greater.
- **TF**. The TF I designed is *c_t_D * ln(1.0 + (k * Lavg) / Ld)*. Since the longer document may have larger chance to have more terms, I use a normalizer *ln(1.0 + (k * Lavg) / Ld)* to compensate the short documents and give penalty to long documents. The final term is the frequency of the term in the current document after normalization. k is a hyperparameter to be tuned. My tuning range is [0,5].
- **QTF**. The QTF I used is the frequency of the term in the query. Generally, if a term occurs more in the query it might be more important, so I directly add the QTF to the formula.
- **CF**. The CF I designed is *(cf+1) / (df * (TF+1))*, where cf is the total frequency of term in the collection. This term is used to put more weight on the term frequency in the collection. If a specific term appears seldom in the collection, the number of the documents we want may be very small since we only want documents with that term. However, other documents may have more matching terms (not the specific term), so the result is inaccurate. The introducing of CF can put more weight to the documents with seldom-used terms.