IR Assignment 1 Report, Group No. 69

Members –
Shristi V Kotaiah, MT22070
Shreeyash Khalate, MT22069
Ujjwal Garg, MT22085

Q1. Data Pre-processing

Methodology:

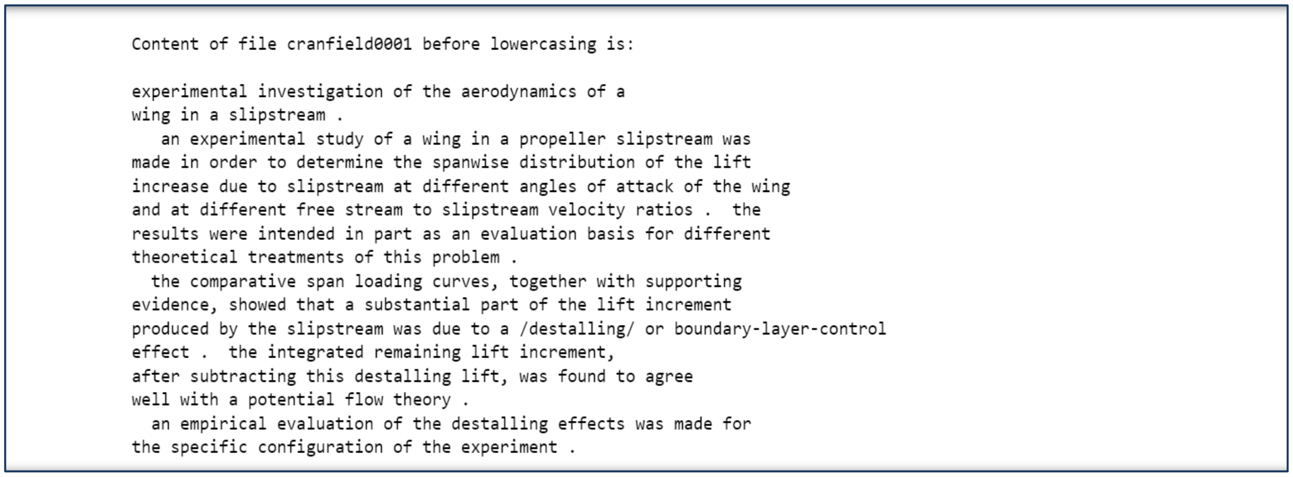
1. Relevant libraries are imported
2. The path of the working directory where dataset lies in the local machine is set, and the working directory is set to that location with 'os' library (os.chdir())
3. Relevant text is extracted between <TITLE></TITLE> and <TEXT></TEXT> tags and the retrieved text is overwritten in the same files.
4. Preprocessing functions are defined and applied on each document and a final preprocessed dataset is stored in a list 'final_dataset'

Preprocessing Steps:

1. The text of each document is lowercased
2. The text of each document in converted into tokens, each token representing one word
3. Stopwords are removed from tokens
4. Punctuations are removed from tokens. Any pattern like [^\w\s] is replaced with whitespaces ' '
5. Blank space tokens are removed

Results:

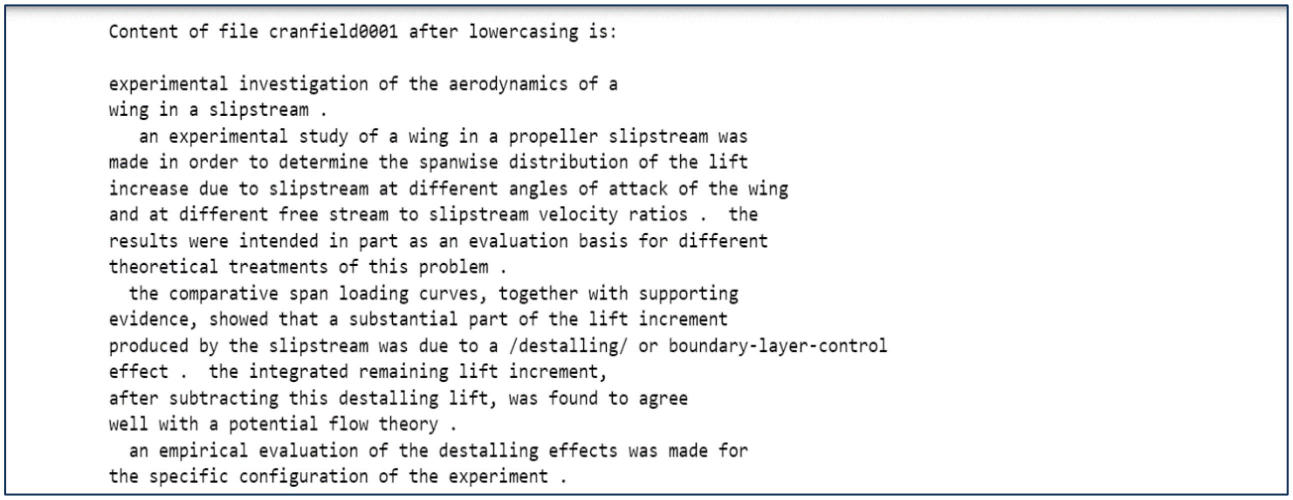
Initial sample document 'cranfield0001'



```
Content of file cranfield0001 before lowercasing is:

experimental investigation of the aerodynamics of a
wing in a slipstream .
   an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios .  the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
  the comparative span loading curves, together with supporting
evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or boundary-layer-control
effect .  the integrated remaining lift increment,
after subtracting this destalling lift, was found to agree
well with a potential flow theory .
  an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .
```

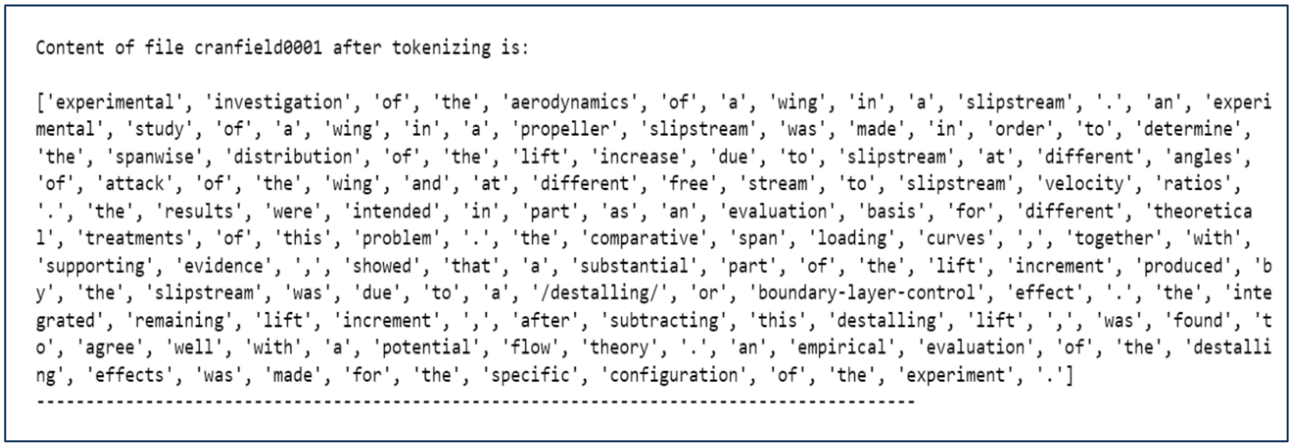
1. After lowercasing the text



```
Content of file cranfield0001 after lowercasing is:

experimental investigation of the aerodynamics of a
wing in a slipstream .
   an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios .  the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
  the comparative span loading curves, together with supporting
evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or boundary-layer-control
effect .  the integrated remaining lift increment,
after subtracting this destalling lift, was found to agree
well with a potential flow theory .
  an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .
```

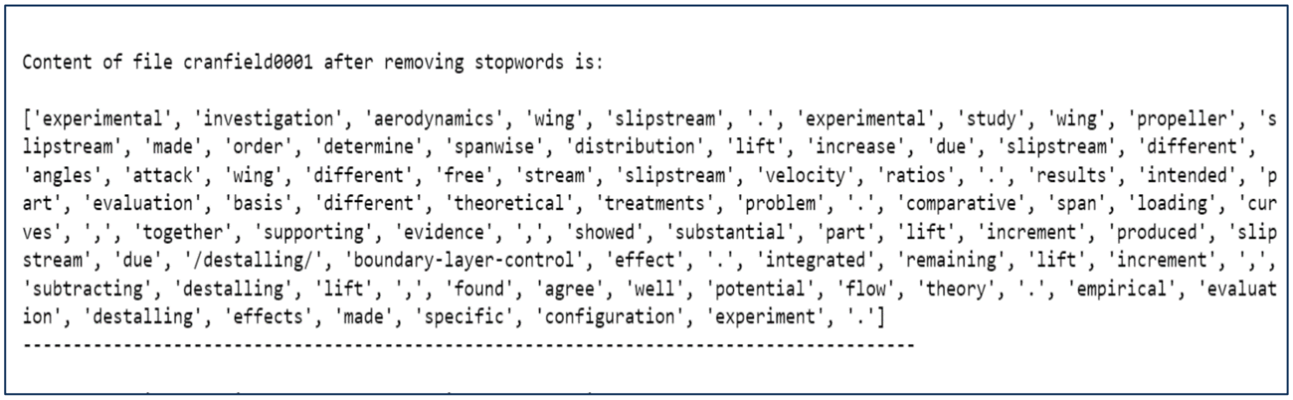
2. After tokenizing the text



```
Content of file cranfield0001 after tokenizing is:

['experimental', 'investigation', 'of', 'the', 'aerodynamics', 'of', 'a', 'wing', 'in', 'a', 'slipstream', '.', 'an', 'experi
mental', 'study', 'of', 'a', 'wing', 'in', 'a', 'propeller', 'slipstream', 'was', 'made', 'in', 'order', 'to', 'determine',
'the', 'spanwise', 'distribution', 'of', 'the', 'lift', 'increase', 'due', 'to', 'slipstream', 'at', 'different', 'angles',
'of', 'attack', 'of', 'the', 'wing', 'and', 'at', 'different', 'free', 'stream', 'to', 'slipstream', 'velocity', 'ratios',
'.', 'the', 'results', 'were', 'intended', 'in', 'part', 'as', 'an', 'evaluation', 'basis', 'for', 'different', 'theoretica
l', 'treatments', 'of', 'this', 'problem', '.', 'the', 'comparative', 'span', 'loading', 'curves', ',', 'together', 'with',
'supporting', 'evidence', ',', 'showed', 'that', 'a', 'substantial', 'part', 'of', 'the', 'lift', 'increment', 'produced', 'b
y', 'the', 'slipstream', 'was', 'due', 'to', 'a', '/destalling/', 'or', 'boundary-layer-control', 'effect', '.', 'the', 'inte
grated', 'remaining', 'lift', 'increment', ',', 'after', 'subtracting', 'this', 'destalling', 'lift', ',', 'was', 'found', 't
o', 'agree', 'well', 'with', 'a', 'potential', 'flow', 'theory', '.', 'an', 'empirical', 'evaluation', 'of', 'the', 'destalli
ng', 'effects', 'was', 'made', 'for', 'the', 'specific', 'configuration', 'of', 'the', 'experiment', '.']
------------------------------------------------------------------------------------------
```

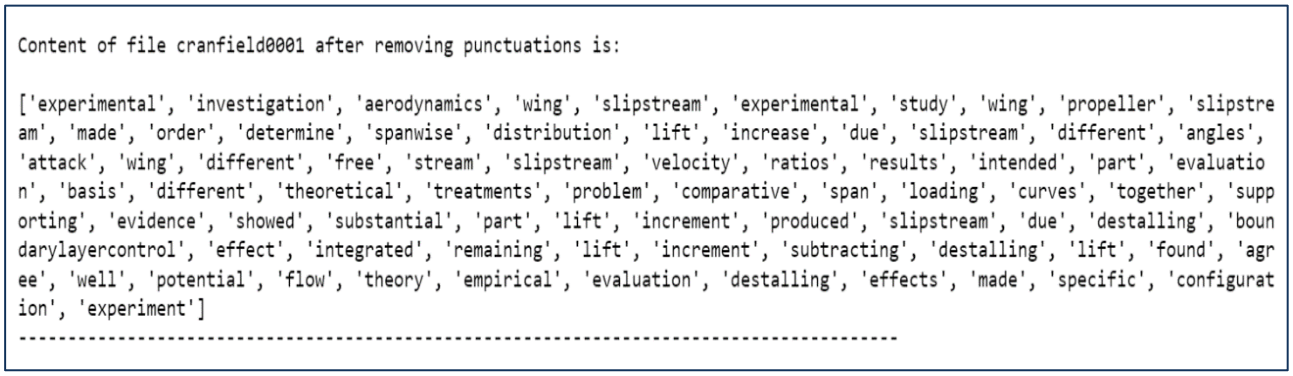
3. After removing stopwords



```
Content of file cranfield0001 after removing stopwords is:

['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', '.', 'experimental', 'study', 'wing', 'propeller', 's
lipstream', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different',
'angles', 'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', '.', 'results', 'intended', 'p
art', 'evaluation', 'basis', 'different', 'theoretical', 'treatments', 'problem', '.', 'comparative', 'span', 'loading', 'cur
ves', ',', 'together', 'supporting', 'evidence', ',', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slip
stream', 'due', '/destalling/', 'boundary-layer-control', 'effect', '.', 'integrated', 'remaining', 'lift', 'increment', ',',
'subtracting', 'destalling', 'lift', ',', 'found', 'agree', 'well', 'potential', 'flow', 'theory', '.', 'empirical', 'evaluat
ion', 'destalling', 'effects', 'made', 'specific', 'configuration', 'experiment', '.']
------------------------------------------------------------------------------------------
```

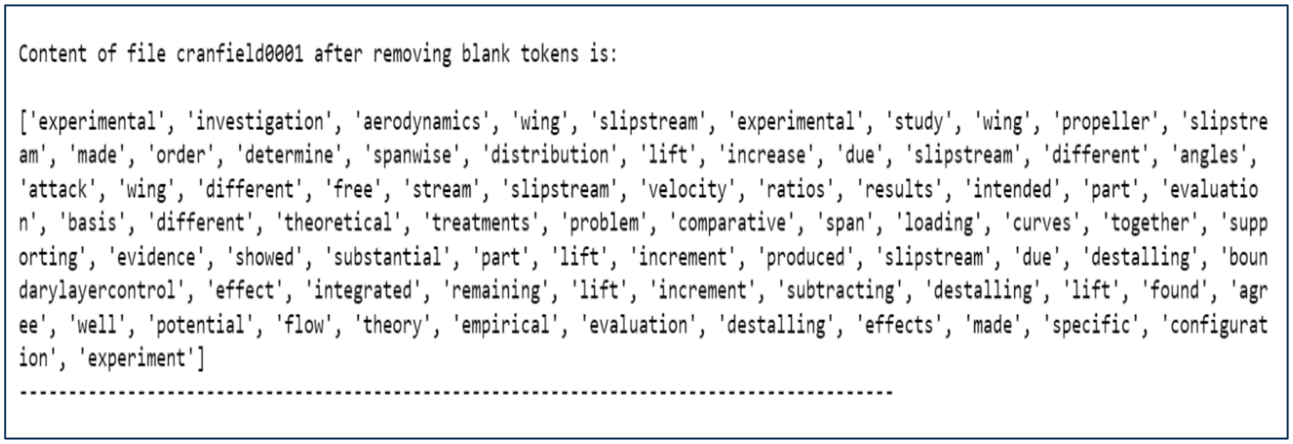
4. After removing punctuations



```
Content of file cranfield0001 after removing punctuations is:

['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', 'experimental', 'study', 'wing', 'propeller', 'slipstre
am', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles',
'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', 'results', 'intended', 'part', 'evaluatio
n', 'basis', 'different', 'theoretical', 'treatments', 'problem', 'comparative', 'span', 'loading', 'curves', 'together', 'supp
orting', 'evidence', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slipstream', 'due', 'destalling', 'boun
darylayercontrol', 'effect', 'integrated', 'remaining', 'lift', 'increment', 'subtracting', 'destalling', 'lift', 'found', 'agr
ee', 'well', 'potential', 'flow', 'theory', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configurat
ion', 'experiment']
------------------------------------------------------------------------------------------
```

5. After removing blank space tokens



```
Content of file cranfield0001 after removing blank tokens is:

['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', 'experimental', 'study', 'wing', 'propeller', 'slipstre
am', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles',
'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', 'results', 'intended', 'part', 'evaluatio
n', 'basis', 'different', 'theoretical', 'treatments', 'problem', 'comparative', 'span', 'loading', 'curves', 'together', 'supp
orting', 'evidence', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slipstream', 'due', 'destalling', 'boun
darylayercontrol', 'effect', 'integrated', 'remaining', 'lift', 'increment', 'subtracting', 'destalling', 'lift', 'found', 'agr
ee', 'well', 'potential', 'flow', 'theory', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configurat
ion', 'experiment']
--------------------------------------------------------------------------------------
```

Assumptions:

Processed data is written back to the same file only is Q1 (i)

For Q1 (ii), after each pre-processing step, the processed data is not written back to the file rather is stored in a separate list.

Q2. Unigram Inverted Index

Methodology:

Creating the unigram inverted index

1. An empty dictionary is created to store unigram inverted indexes
2. Each document token list in the preprocessed corpus dataset is iterated
3. Tokens for each document are created in each iteration
4. For each token created, if that token already exists in dictionary, then corresponding document is added in the value component of that token in the dictionary
5. If token does not exist in the dictionary, then a new key-value pair (token-document) is added in the dictionary
6. Unigram index dictionary is dumped in local directory using 'pickle' and loaded back

Boolean query processing

1. All the pre-processing steps of Q1 are applied to input query by calling appropriate functions sequentially
2. A get_not() function is defined which retrieves all the documents in the corpus that does not contain the passed word. Here comparisons will be equal to total documents in the corpus.
3. A generate_operators_operands() function separates the input query words and operators (and, or) in separate lists
4. A set intersection() and set union() functions are defined to take intersection and union of two passed sets. Here comparisons will be equal to number of documents in set1. (Not optimized for iterating the set with less number of documents)
5. The and_or() function takes the intersection or union of the postings set of each

query token in a sequential manner

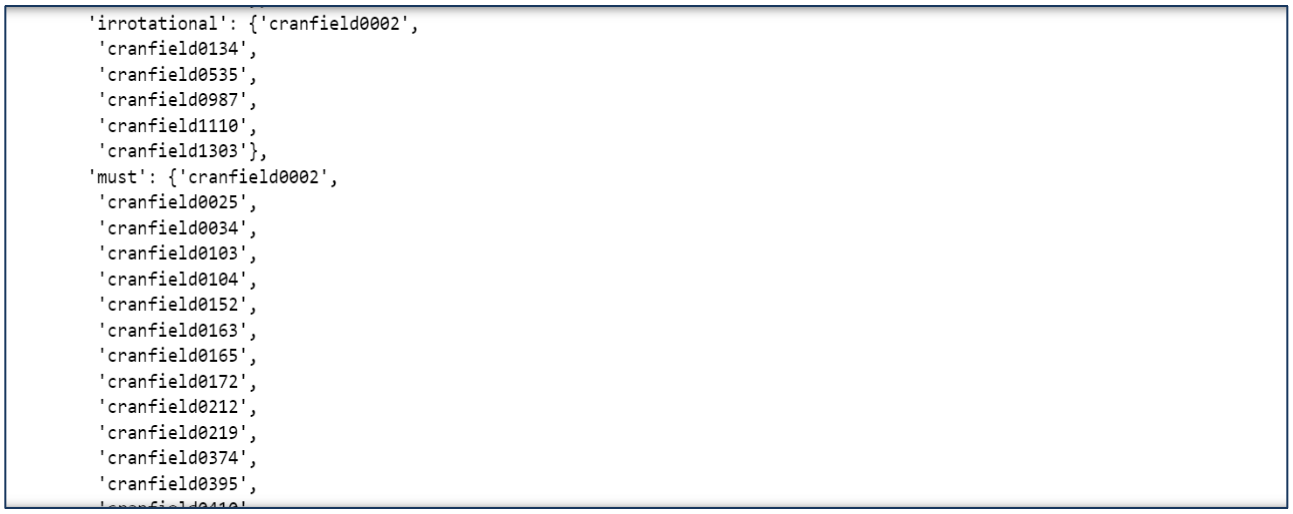
6. The run_query() function processes all ‘not’ operators first and stores their corresponding returned sets in separate list ‘not_set_list’ before taking intersection or union depending on ‘and’ or ‘or’ operation.
7. Note that each term of the type ‘NOT term’ is first processed and corresponding not set index in not_set_list is passed in operand in place of ‘not term’

Assumptions:

1. The number of tokens present in the processed query tokens list must be exactly 1 more than the number of operators passed for that query.
2. This does not apply for input query containing only one token and one not operator. In this case, program automatically returns appropriate document set for given query of type ‘NOT token’
3. If point 1 is not satisfied for queries of type other than the type mentioned in point 2, then the program will automatically scale down the number of tokens if not enough operators are provided or it will scale down number of operators if not enough tokens are available to apply those operators on. Scaling down here means, the extra operands or operators will be ignored

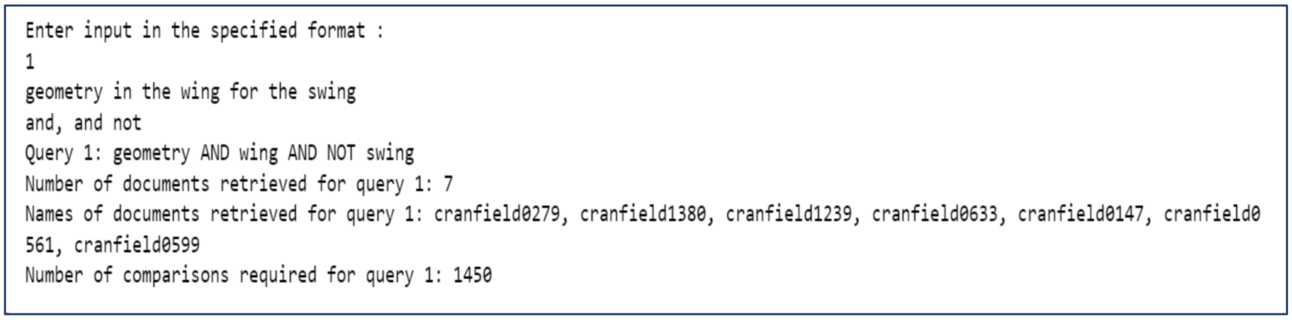
Results:

Unigram Inverted Index -



```
 'irrotational': {'cranfield0002',
  'cranfield0134',
  'cranfield0535',
  'cranfield0987',
  'cranfield1110',
  'cranfield1303'},
 'must': {'cranfield0002',
  'cranfield0025',
  'cranfield0034',
  'cranfield0103',
  'cranfield0104',
  'cranfield0152',
  'cranfield0163',
  'cranfield0165',
  'cranfield0172',
  'cranfield0212',
  'cranfield0219',
  'cranfield0374',
  'cranfield0395',
```

Sample Query Input –



```
Enter input in the specified format :
1
geometry in the wing for the swing
and, and not
Query 1: geometry AND wing AND NOT swing
Number of documents retrieved for query 1: 7
Names of documents retrieved for query 1: cranfield0279, cranfield1380, cranfield1239, cranfield0633, cranfield0147, cranfield0
561, cranfield0599
Number of comparisons required for query 1: 1450
```

Q3. Bigram Inverted Index and Positional Index

Methodology:

(i) Creating the Bigram inverted index

1. An empty dictionary is created to store bigram inverted indexes
2. Each document token list in the preprocessed corpus dataset is iterated
3. Tokens for each document are created in each iteration
4. For each ith token, a biword is created in the form "token[i]+' '+token[i+1]" until the (n-1)th token where n is number of tokens in current document
5. If that biword already exists in dictionary, then corresponding document is added in the value component (posting) of that byword in the dictionary
6. If biword does not exist in the dictionary, then a new key-value pair (biword-document) is added in the dictionary
7. Bigram index dictionary is dumped in local directory using 'pickle' and loaded back

(ii) Creating the Positional index

1. An empty dictionary is created to store positional indexes
2. Each document token list in the preprocessed corpus dataset is iterated
3. Tokens for each document are created in each iteration and each token is iterated
4. If that token is positional index dictionary and also its docID is in
5. If that biword already exists in dictionary and also its docID is in postings list of that token, then we update the token position in postings list of docID for that token
6. If token exists but its docID does not, then new docID is added in postings list of that token along with its position
7. If token is not present at all in the dictionary, then that token with its docID and position is added in the positional index dictionary
8. Final positional index dictionary is dumped in the local folder and loaded back using 'pickle'

(iii) Boolean query processing

1. All the pre-processing steps of Q1 are applied to input query by calling appropriate functions sequentially
2. A run_query_bigram() function iterates through all input query tokens (sanitized) and creates a biword for each ith and (i+1)th token.
3. In initial iteration, the result set consists the postings list of the very first biword. From the 2nd iteration onwards, the intersection of postings list set of current biword and previous result set is taken and stored back in result set variable. In this way, result set after last iteration is out required set of documents.

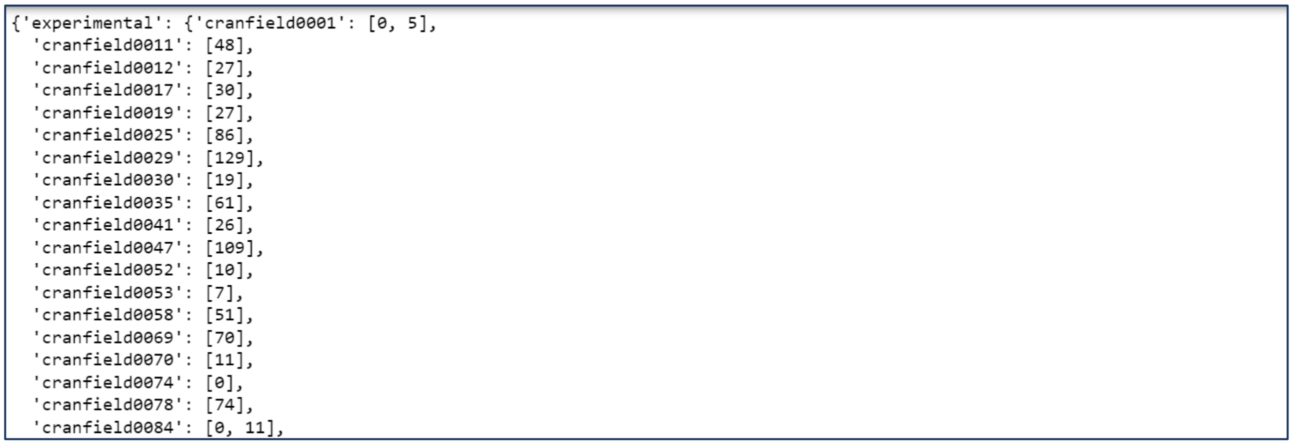
4. The helper function positional_find_matching_documents(tokens) finds all the documents in which all passed query tokens are present.
5. The helper function positional_intersect_for_token_pair(token1, token2, distance) returns the set of documents in which token2 is farther than 'distance' words from the token1
6. A run_query_positional(query_tokens) function first retrieves the matching documents i.e. the documents in which all input query tokens (sanitized) are present. Is this matching documents sets itself is empty then no need to look further as our resulting set will be empty set (no need to check relative positions of tokens)
7. If matching documents set is not empty, then we check for documents satisfying relatives positions of each pair of tokens, by intersecting the document set returned from positional_intersect_for_token_pair() for each token pair with previous result set and storing the result back in result set.
8. We consider each token pair by nested for loop where,
   ***i** goes from 0 to len(query_tokens)* and ***j** goes from (i+1) to len(query_tokens)*
   For each such iteration, the distance to check between two tokens will be,
   |(j-1)|

Assumptions:

1. The document containing text : "…token1 stopword1 stopword2 stopword3 token2…" is considered valid for phrase query "token1 stopword4 stopword5 token2". These cases won't count in False Positives as we perform preprocessing on documents as well as on the input query. Thus stopwords won't make any impact.
2. The relative positions of tokens with each other are taken into consideration only after pre-processing the input query and NOT before, as we need to apply these token positions on an already pre-processed data

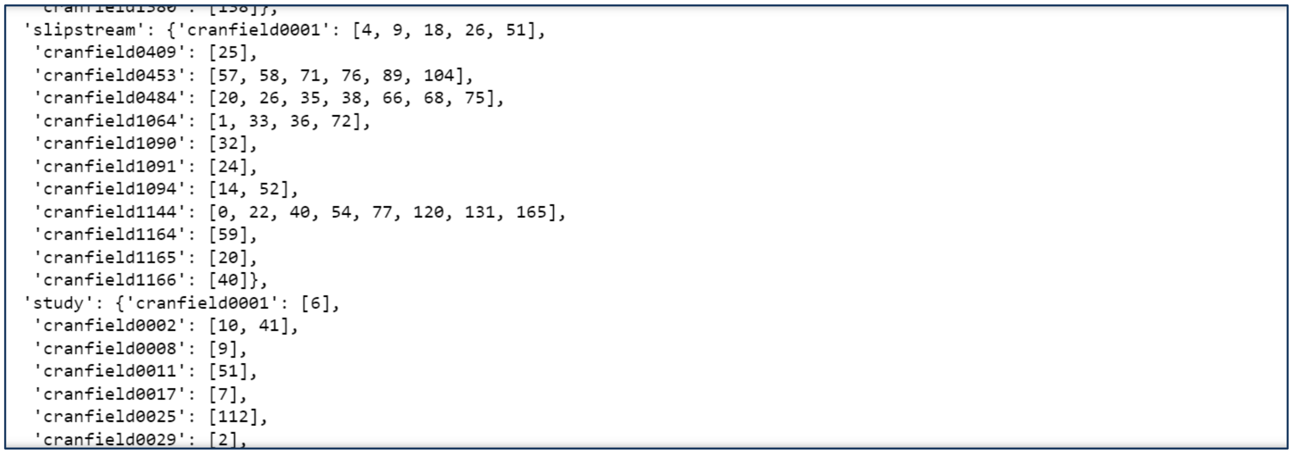
Results:

Bigram Inverted Index -

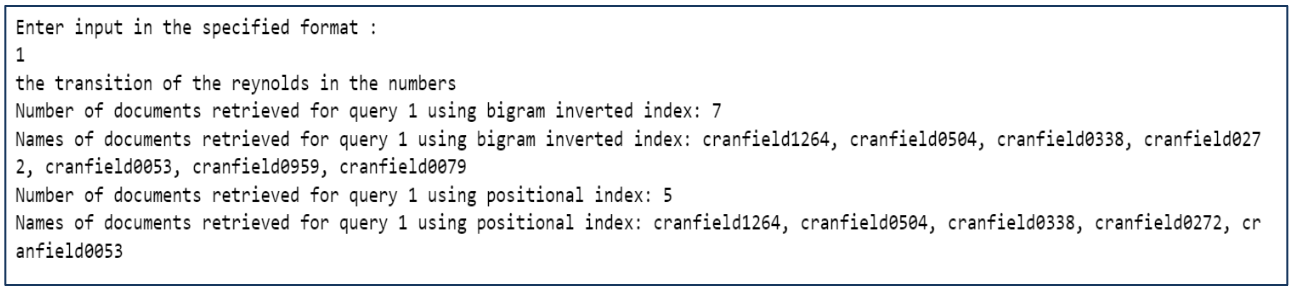


```
{'experimental': {'cranfield0001': [0, 5],
  'cranfield0011': [48],
  'cranfield0012': [27],
  'cranfield0017': [30],
  'cranfield0019': [27],
  'cranfield0025': [86],
  'cranfield0029': [129],
  'cranfield0030': [19],
  'cranfield0035': [61],
  'cranfield0041': [26],
  'cranfield0047': [109],
  'cranfield0052': [10],
  'cranfield0053': [7],
  'cranfield0058': [51],
  'cranfield0069': [70],
  'cranfield0070': [11],
  'cranfield0074': [0],
  'cranfield0078': [74],
  'cranfield0084': [0, 11],
```

Positional Index -

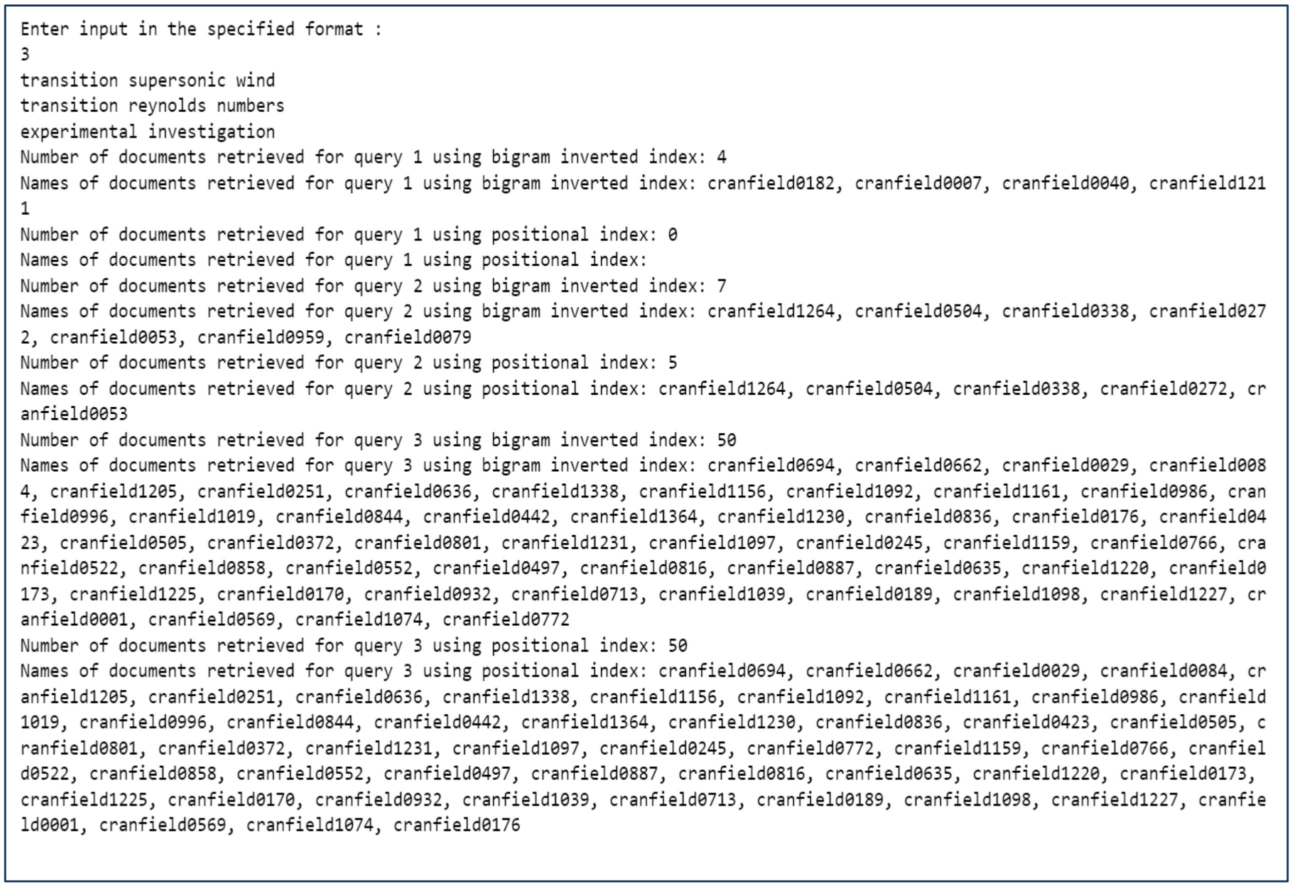

```
  'cranfield1380': [138]},
 'slipstream': {'cranfield0001': [4, 9, 18, 26, 51],
  'cranfield0409': [25],
  'cranfield0453': [57, 58, 71, 76, 89, 104],
  'cranfield0484': [20, 26, 35, 38, 66, 68, 75],
  'cranfield1064': [1, 33, 36, 72],
  'cranfield1090': [32],
  'cranfield1091': [24],
  'cranfield1094': [14, 52],
  'cranfield1144': [0, 22, 40, 54, 77, 120, 131, 165],
  'cranfield1164': [59],
  'cranfield1165': [20],
  'cranfield1166': [40]},
 'study': {'cranfield0001': [6],
  'cranfield0002': [10, 41],
  'cranfield0008': [9],
  'cranfield0011': [51],
  'cranfield0017': [7],
  'cranfield0025': [112],
  'cranfield0029': [2],
```

Sample Input Query -


```
Enter input in the specified format :
1
the transition of the reynolds in the numbers
Number of documents retrieved for query 1 using bigram inverted index: 7
Names of documents retrieved for query 1 using bigram inverted index: cranfield1264, cranfield0504, cranfield0338, cranfield027
2, cranfield0053, cranfield0959, cranfield0079
Number of documents retrieved for query 1 using positional index: 5
Names of documents retrieved for query 1 using positional index: cranfield1264, cranfield0504, cranfield0338, cranfield0272, cr
anfield0053
```

Comparison of Results using Bigram Inverted Index and Positional Index:

Another Sample Input Query -


```
Enter input in the specified format :
3
transition supersonic wind
transition reynolds numbers
experimental investigation
Number of documents retrieved for query 1 using bigram inverted index: 4
Names of documents retrieved for query 1 using bigram inverted index: cranfield0182, cranfield0007, cranfield0040, cranfield121
1
Number of documents retrieved for query 1 using positional index: 0
Names of documents retrieved for query 1 using positional index:
Number of documents retrieved for query 2 using bigram inverted index: 7
Names of documents retrieved for query 2 using bigram inverted index: cranfield1264, cranfield0504, cranfield0338, cranfield027
2, cranfield0053, cranfield0959, cranfield0079
Number of documents retrieved for query 2 using positional index: 5
Names of documents retrieved for query 2 using positional index: cranfield1264, cranfield0504, cranfield0338, cranfield0272, cr
anfield0053
Number of documents retrieved for query 3 using bigram inverted index: 50
Names of documents retrieved for query 3 using bigram inverted index: cranfield0694, cranfield0662, cranfield0029, cranfield008
4, cranfield1205, cranfield0251, cranfield0636, cranfield1338, cranfield1156, cranfield1092, cranfield1161, cranfield0986, cran
field0996, cranfield1019, cranfield0844, cranfield0442, cranfield1364, cranfield1230, cranfield0836, cranfield0176, cranfield04
23, cranfield0505, cranfield0372, cranfield0801, cranfield1231, cranfield1097, cranfield0245, cranfield1159, cranfield0766, cra
nfield0522, cranfield0858, cranfield0552, cranfield0497, cranfield0816, cranfield0887, cranfield0635, cranfield1220, cranfield0
173, cranfield1225, cranfield0170, cranfield0932, cranfield0713, cranfield1039, cranfield0189, cranfield1098, cranfield1227, cr
anfield0001, cranfield0569, cranfield1074, cranfield0772
Number of documents retrieved for query 3 using positional index: 50
Names of documents retrieved for query 3 using positional index: cranfield0694, cranfield0662, cranfield0029, cranfield0084, cr
anfield1205, cranfield0251, cranfield0636, cranfield1338, cranfield1156, cranfield1092, cranfield1161, cranfield0986, cranfield
1019, cranfield0996, cranfield0844, cranfield0442, cranfield1364, cranfield1230, cranfield0836, cranfield0423, cranfield0505, c
ranfield0801, cranfield0372, cranfield1231, cranfield1097, cranfield0245, cranfield0772, cranfield1159, cranfield0766, cranfiel
d0522, cranfield0858, cranfield0552, cranfield0497, cranfield0887, cranfield0816, cranfield0635, cranfield1220, cranfield0173,
cranfield1225, cranfield0170, cranfield0932, cranfield1039, cranfield0713, cranfield0189, cranfield1098, cranfield1227, cranfie
ld0001, cranfield0569, cranfield1074, cranfield0176
```

As we can see from above query inputs, For query 1 - 'transition supersonic wind', the bigram inverted index shows 4 documents which says that 'transition supersonic wind' is present in these 4 documents. Although these words do appear in these documents but they do not appear in the specified sequence and adjacent to each other. Hence Bigram inverted index shows **False Positives** for our query. Positional index retrieves 0 documents for the same query. Thus positional index is a better representation of document indexes for queries containing more than 2 words as Positional index also takes into account the relative positions of the input query words and Bigram index doesn't.

Also, in query 2 - 'transition reynolds numbers', the bigram inverted index shows 7 documents while positional index shows just 5 documents. The document set shown by positional index is the true set as 'transition reynolds numbers' appear in the exact given sequence in only these 5 documents (ignoring the stopwords). Thus Bigram index shows two false positives in this case. Thus we can conclude that, the number of documents retrieved by Positional index will always be **less than or equal to** number of documents retrieved by Bigram inverted index. Bigram inverted index will often show **False Positives** for queries with more than 2 tokens

On the other hand, Bigram index and Positional index shows same result for query 3 - 'experimental investigation'. This is because for a 2-word input query (after removing stopwords), Bigram index and Positional index behaves same. Not just for this particular input but for any input query containing only 2 tokens, the Bigram and Positional index will always show the same result as a 2 word query in positional index just behaves same a biword behaves in bigram index. In Positional index, the fact that position of token2 is 1 ahead of position of token1 is considered. And bigram index is constructed on basis of this very same assumption. Thus for a 2-word query searching, bigram inverted index and positional index will always show same result.