We perform the task given in the following 3 steps

- We first perform preprocessing on the given dataset and store all the terms extracted
- Then we form an inverted index where each file is assigned a unique number.
- Then we implement different operations such as AND, OR, AND NOT and OR NOT.

Q1 a Carry out the suitable preprocessing steps on the given dataset

- First step involves noise removal which includes removal of file headers, footers,markup data and extra spaces.
- Second step involves replacing contractions with their expansions, which should be done before tokenization.
- Third step involves tokenization using NLTK's word_tokenize()
- Fourth Step involves removal of non-ascii characters from the list of tokenized words.
- Fifth step involves conversion of each character to lowercase from list of tokenized words.
- Sixth step involves removal of punctuations from the list of tokenized words.
- Seventh step involves removal of stop words from the list of tokenized words.
- The Final step involves lemmatization, we tried with wordnet lemmatizer (nltk) along with pos tags and other one is using spacy lemmatizer.
- For all the documents we generated a dictionary, where document name is the key and value is the list of preprocessed words of that document, which is used for generating inverted index.

Q2 : Creating unigram inverted index.

- This step involves creating a dictionary (inverted index) containing all the unique terms present inside all the given documents along with a posting list for every term. The list corresponding to a particular term holds the document id of the documents where the term is present.
- The input through the pre-processing step is a dictionary, the document name as the key and the text inside that document as the value. The text inside is pre-processed. As part of the process, every document is given a unique document id.
- The text inside every document is read, if a particular term is not present in the invertex index, the term is added and the document id of the document in which it is present is also added. If the term is present then only the document id is added to the posting list of that term.
- The inverted index contains only unique terms, but a term can occur in a document many times, which leads to addition of the same document id in the posting list of a particular term multiple times. So from the posting list of each term, duplicates are removed.
- The posting lists are sorted in increasing order of document ids for reducing the number of comparisons.
- Finally, the document frequency is calculated based on the size of the posting list, for each term. Document frequency for a term indicates the number of documents in which the term is present. Document frequency helps in reducing the number of comparisons.

Query processing

- Suitable preprocessing systems as explained above for the documents are applied on the input query too
- The user is displayed the processed query to ensure the number of operators match the required terms
- The operation is applied sequentially from left to right, with one subsequent term each time
- The number of **comparisons**, number of **documents matched**, and the **retrieved documents** are displayed immediately after each query.

Sample outputs:
Given the query - “lion stood thoughtfully for a moment”

```
Enter the query with terms separated by spacelion stood thoughtfully for a moment
query terms lion stand thoughtfully moment

Enter the operation sequence separated by commaOR,OR,or
Number of comparisons 822
Number of documents matched 311
The retrieved documents are: [aluminum.hum, sight.txt, beyond.hum, partya.txt, nitepeek.sto, blind.txt, altside.hum, abyss.txt,
t_zone.jok, advsayed.txt, eyeargon.hum, elite.app, rocket.sf, corcor.hum, ab40thv.txt, gold3ber.txt, game.txt, excerpt.txt,
knuckle.txt, emperor3.txt, empnclot.txt, abbey.txt, archive, immorti.hum, wlgirl.txt, cooldark.sto, vday.hum, imagin.hum,
adv_alad.txt, testpilo.hum, valen, enchdup.hum, ladylust.hum, tcoa.txt, enginer.txt, korea.s, taxnovel.txt, greedog.txt,
keepmodu.txt, vainsong.txt, goldgoos.txt, goldbug.poe, grav, empty.txt, gulliver.txt, gay, ghost, goldfish.txt, tao3.dos,
enya_trn.txt, ezoff, girlclub.txt, gatherng.txt, girl, enc, qcarroll, goldenp.txt, greatlrn.leg, yukon.txt, veiledl.txt,
unluckwr.txt, lgoldbrd.txt, aquith.txt, beast.asc, bulzork1.txt, bulironb.txt, bureau.txt, beautbst.txt, bulphrek.txt, bgcspoof.txt,
bulfelis.txt, burintrv.66, burintrv.92, mindprob.txt, lament.txt, sucker.txt, angry_ca.txt, zombies.txt, wisteria.txt, outcast.dos,
quest, withdraw.cyb, beggars.txt, buggy.txt, blue, bishop00.txt, blackp.txt, bulhuntr.txt, blh.txt, buldream.txt, bulmrx.txt,
blackrdr, blak, bulprint.txt, bulolli1.txt, bulnland.txt, bulolli2.txt, graymare.txt, batlslau.txt, bluebrd.txt, burn, s&m_plot,
s&m_that, arcadia.sty, tuc_mees, sanpedr2.txt, mattress.txt, mazarin.txt, tearglas.txt, timetrav.txt, 13chil.txt, 14.lws, 16.lws,
18.lws, 19.lws, 20.lws, 5orange.txt, 6napolen.txt, 7voysinb.txt, musgrave.txt, jackbstl.txt, 3gables.txt, 3lpigs.txt, 3student.txt,
radar_ra.txt, rainda.txt, reap, shoscomb.txt, shrdfarm.txt, shulk.txt, sick-kid.txt, silverb.txt, sis, sleprncs.txt, snowmaid.txt,
snowqn1.txt, piracy.sto, paul_har.sto, hareporc.txt, haretort.txt, stainles.ana, angelfur.hum, blabnove.hum, blabnove.txt,
brain.damage, bulwer.lytton, crazy.hum, excerpt.hum, fantasy.hum, fantasy.txt, fred.txt, hitch2.txt, hitch3.txt, hotline1.txt,
hotline3.txt, hotline4.txt, idi.hum, jerichms.hum, dan, darkness.txt, deal, bran, breaks1.asc, breaks2.asc, breaks3.asc, bruce-
```

Showing different query formats:

```
Enter number of queries4

Enter the query with terms separated by spaceboom bam
query terms boom bam

Enter the operation sequence separated by commaAND
Number of comparisons 26
Number of documents matched 1
The retrieved documents are: [times.fic, ]

Enter the query with terms separated by spaceboom bam
query terms boom bam

Enter the operation sequence separated by commaOR
Number of comparisons 26
Number of documents matched 28
The retrieved documents are: [rocket.sf, empty.txt, beast.asc, bureau.txt, bgcspoof.txt, bulmrx.txt, 7voysinb.txt, sick-kid.txt,
blabnove.hum, blabnove.txt, breaks1.asc, breaks2.asc, floc, pinocch.txt, cabin.txt, sqzply.txt, cybersla.txt, hound-b.txt, fic1,
non4, history5.txt, times.fic, fourth.fic, robotech, hellmach.txt, bookem2, assorted.txt, forgotte, ]

Enter the query with terms separated by spaceboom bam
query terms boom bam

Enter the operation sequence separated by commaAND NOT
Number of comparisons 26
Number of documents matched 24
The retrieved documents are: [rocket.sf, empty.txt, bureau.txt, bgcspoof.txt, bulmrx.txt, 7voysinb.txt, sick-kid.txt, blabnove.hum,
blabnove.txt, breaks1.asc, breaks2.asc, floc, pinocch.txt, cabin.txt, sqzply.txt, cybersla.txt, hound-b.txt, fic1, history5.txt,
fourth.fic, robotech, hellmach.txt, assorted.txt, forgotte, ]
```



```
Enter the operation sequence separated by commaOR NOT
Number of comparisons 419
Number of documents matched 464
The retrieved documents are: [aluminum.hum, life.txt, sight.txt, cameloto.hum, beyond.hum, partya.txt, nitepeek.sto, blind.txt,
altside.hum, abyss.txt, t_zone.jok, fantas.hum, advsayed.txt, eyeargon.hum, elite.app, rocket.sf, corcor.hum, elveshoe.txt,
ab40thv.txt, gold3ber.txt, game.txt, excerpt.txt, knuckle.txt, emperor3.txt, empnclot.txt, abbey.txt, advtthum.txt, archive,
wolfcran.txt, immorti.hum, wlgirl.txt, cooldark.sto, wolf7kid.txt, vday.hum, imagin.hum, adv_alad.txt, wombat.und, gemdra.txt,
aircon.txt, wolflamb.txt, testpilo.hum, valen, confilct.fun, narciss.txt, enchdup.hum, ladylust.hum, tcoa.txt, enginer.txt,
encamp01.txt, korea.s, taxnovel.txt, greedog.txt, keepmodu.txt, vainsong.txt, goldgoos.txt, quickfix, goldbug.poe, omarsheh.txt,
grav, empty.txt, oxfrog.txt, gulliver.txt, obstgoat.txt, gay, ghost, goldfish.txt, tao3.dos, empsjowk.txt, enya_trn.txt, quot,
kharian.txt, ezoff, girlclub.txt, tctac.txt, gatherng.txt, tailbear.txt, traitor.txt, gloves.txt, vampword.txt, girl, vaincrow.txt,
```

When no documents match the query:

```
Enter number of queries1

Enter the query with terms separated by spaceboom asd
query terms boom asd

Enter the operation sequence separated by commaAND
Number of comparisons 12
Number of documents matched 0
The retrieved documents are: []
```

When term is not found in any document for an “AND” operator query:

```
Enter number of queries1

Enter the query with terms separated by spaceasdf boom
query terms asdf boom

Enter the operation sequence separated by commaAND
Term not found: asdf
Hence, operation has failed
```

b) Implement the unigram inverted index data structure.
Approach:

c) Implement the following operations:

- X OR Y

  Approach:

  This is equivalent to the union operation. We aim to perform the union of two postings. The algorithm can be described as follows.

  OR(p1,p2)

  1. answer <-[]
  2. While p1!= NIL and p2!=NIL
     a. If docid[p1]==docid[p2]:
     b. append(answer,docid[p1])
     c. p1->p1.next
     d. p2->p2.next
     e. Elif docid[p1]<docid[p2]
     f. append(answer,docid[p1])
     g. p1->p1.next
     h. Else
     i. append(answer,docid(p2))
     j. p2->p2.next
  3. While p1!=NIL:
     append(answer,docid(p1))
  4. While p2!=NIL:
     a. append(answer,docid(p2))
  5. Return answer

  Algorithm description:

  For OR operator we perform an union of the postings. The postings are sorted in ascending order. We traverse both the posting lists and add the docId if the docIDs in both the postings are equal and the iteration through the list continues. We also add the respective docIDS even if they are not equal to each other then the counter pointing to the smaller docID is advanced. Then for merging the rest of the items in postings we iterate through the list which has not still been exhausted and add the items to the answer list. The answer list is then returned.

Example outputs for OR:

```
Enter number of queries1
Enter the query with terms separated by spacelion stood thoughtfully for a moment
query terms lion stand thoughtfully moment
Enter the operation sequence separated by commaOR,OR,OR
Number of comparisons 822
Number of documents matched 311
The retrieved documents are: [aluminum.hum, sight.txt, beyond.hum, partya.txt, nitepeek.sto, blind.txt, altside.hum, abyss.txt, t_zone.jok, advsayed.txt, eyeargon.hum,
 elite.app, rocket.sf, corcor.hum, ab40thv.txt, gold3ber.txt, game.txt, excerpt.txt, knuckle.txt, emperor3.txt, empnclot.txt, abbey.txt, archive, immorti.hum, wlgirl.t
xt, cooldark.sto, vday.hum, imagin.hum, adv_alad.txt, testpilo.hum, valen, enchdup.hum, ladylust.hum, tcoa.txt, enginer.txt, korea.s, taxnovel.txt, greedog.txt, keepmo
du.txt, vainsong.txt, goldgoos.txt, goldbug.poe, grav, empty.txt, gulliver.txt, gay, ghost, goldfish.txt, tao3.dos, enya_trn.txt, ezoff, girlclub.txt, gatherng.txt, gi
rl, enc, qcarroll, goldenp.txt, greatlrn.leg, yukon.txt, veiledl.txt, unluckwr.txt, lgoldbrd.txt, aquith.txt, beast.asc, bulzork1.txt, bulironb.txt, bureau.txt, beautb
st.txt, bulphrek.txt, bgcspoof.txt, bulfelis.txt, burintrv.66, burintrv.92, mindprob.txt, lament.txt, sucker.txt, angry_ca.txt, zombies.txt, wisteria.txt, outcast.dos,
 quest, withdraw.cyb, beggars.txt, buggy.txt, blue, bishop00.txt, blackp.txt, bulhuntr.txt, blh.txt, buldream.txt, bulmrx.txt, blackrdr, blak, bulprint.txt, bulolli1.t
xt, bulnland.txt, bulolli2.txt, graymare.txt, batlslau.txt, bluebrd.txt, burn, s&m_plot, s&m_that, arcadia.sty, tuc_mees, sanpedr2.txt, mattress.txt, mazarin.txt, tear
glas.txt, timetrav.txt, 13chil.txt, 14.lws, 16.lws, 18.lws, 19.lws, 20.lws, 5orange.txt, 6napolen.txt, 7voysinb.txt, musgrave.txt, jackbstl.txt, 3gables.txt, 3lpigs.tx
t, 3student.txt, radar_ra.txt, rainda.txt, reap, shoscomb.txt, shrdfarm.txt, shulk.txt, sick-kid.txt, silverb.txt, sis, sleprncs.txt, snowmaid.txt, snowqn1.txt, piracy
.sto, paul_har.sto, hareporc.txt, haretort.txt, stainles.ana, angelfur.hum, blabnove.hum, blabnove.txt, brain.damage, bulwer.lytton, crazy.hum, excerpt.hum, fantasy.hu
m, fantasy.txt, fred.txt, hitch2.txt, hitch3.txt, hotline1.txt, hotline3.txt, hotline4.txt, idi.hum, jerichms.hum, dan, darkness.txt, deal, bran, breaks1.asc, breaks2.
asc, breaks3.asc, bruce-p.txt, lil, lionmane.txt, lionmosq.txt, lionwar.txt, lmtchgrl.txt, startrek.txt, deer.txt, descent.poe, diaryflf.txt, long1-3.txt, lrrhood.txt,
 ltp, lure.txt, floc, floobs.txt, flytrunk.txt, paink-ws.txt, perf, mtinder.txt, monkking.txt, monksol.txt, mouslion.txt, missing.txt, pussboot.txt, pinocch.txt, foxnc
row.txt, cardcnt.txt, domain.poe, dopedenn.txt, dskool.txt, dtruck.txt, dwar, redragon.txt, retrib.txt, rock, roger1.txt, running.txt, sunday.txt, superg1, stairdre.tx
t, stsgreek, igiv, immortal, inter, adler.txt, aesop11.txt, aesopa10.txt, alad10.txt, healer.txt, space.txt, spectacl.poe, sqzply.txt, sre-dark.txt, szechuan, solitary
.txt, pregn.txt, psf.txt, psi, psyc, plescopm.txt, cybersla.txt, holmesbk.txt, home.fil, hop-frog.poe, horsdonk.txt, horswolf.txt, hound-b.txt, fic1, fic2, fic3, fic4,
 fic5, fic7, fish.txt, island.poe, foxngrap.txt, fran, fgoose.txt, freeman.fil, friend.s, friends.txt, nigel.10, nigel.2, nigel.3, nigel.5, nigel.6, nihgel_8.9, 4moons
.txt, hils, history5.txt, poplstrm.txt, pphamlin.txt, prince.art, hell4.txt, charlie.txt, chik, clon, cmoutmou.txt, cum, wall.art, blasters.fic, jackmac.fic, reality.t
xt, fourth.fic, campfire.txt, aislesix.txt, mcdonaldl.txt, pepdegener.txt, socialvikings.txt, cooldark.txt, aisle.six, day.in.mcdonald, pepsi.degenerat, social.vikings
, spam.key, textfile.primer, robotech, hellmach.txt, bookem2, bookem.1, bookem3, arctic.txt, bestwish, forgotte, quarter.c1, quarter.c13, quarter.c17, quarter.c18, qua
rter.c19, quarter.c3, vgilante.txt, sre02.txt, sre07.txt, sre10.txt, sre09.txt, sre06.txt, sre_finl.txt, srex.txt, sre04.txt, tree.txt, consumdr.hum, snow.txt, candle.
hum, spiders.txt, timem.hac, ]
```

From the above we said that union of postings is returned. Please note that our answer does not match with the one mentioned in the given assignment sheet example as we have lemmatized the terms in the corpus and the query resulting in a change in number of results.

```
Enter number of queries1
Enter the query with terms separated by spacelion zebra horse
query terms lion zebra horse
Enter the operation sequence separated by commaOR,OR
Number of comparisons 113
Number of documents matched 103
The retrieved documents are: [advsayed.txt, eyeargon.hum, rocket.sf, ab40thv.txt, abbey.txt, archive, wlgirl.txt, valen, enchdup.hum, enginer.txt, korea.s, keepmodu.tx
t, vainsong.txt, omarsheh.txt, grav, gulliver.txt, tao3.dos, empsjowk.txt, girlclub.txt, tailbear.txt, goldenp.txt, greatlrn.leg, yukon.txt, veiledl.txt, weeprncs.txt,
 aminegg.txt, aquith.txt, bureau.txt, beautbst.txt, angry_ca.txt, wisteria.txt, outcast.dos, beggars.txt, bulmrx.txt, blossom.pom, bluebrd.txt, mazarin.txt, melissa.tx
t, 13chil.txt, 14.lws, 16.lws, 18.lws, 6ablemen.txt, 7oldsamr.txt, 7voysinb.txt, radar_ra.txt, shoscomb.txt, sick-kid.txt, silverb.txt, sleprncs.txt, snowqn1.txt, bigr
ed.hum, blabnove.hum, blabnove.txt, hitch2.txt, idi.hum, dakota.txt, darkness.txt, bruce-p.txt, lionmane.txt, lionmosq.txt, lionwar.txt, diaryflf.txt, dicksong.txt, lt
p, lure.txt, perf, monkking.txt, mouslion.txt, missing.txt, pussboot.txt, dskool.txt, dtruck.txt, redragon.txt, aesop11.txt, aesopa10.txt, alad10.txt, fable.txt, sqzpl
y.txt, solitary.txt, psf.txt, cybersla.txt, holmesbk.txt, home.fil, horsdonk.txt, horswolf.txt, hound-b.txt, fic5, fgoose.txt, friends.txt, history5.txt, chik, blaster
s.fic, socialvikings.txt, social.vikings, robotech, bookem2, arctic.txt, forgotte, quarter.c18, vgilante.txt, consumdr.hum, timem.hac, ]
```

Above screenshot shows an example of an OR operator (lion OR zebra OR horse). We do not perform optimization of merging terms with lesser number of postings first as the instructions mention the operations should be performed from left to right. However, we store the document frequency at the head and the optimization can be incorporated easily.

- X AND Y

Approach:

This is equivalent to the intersection operation. The algorithm can be described as follows.

AND(p1,p2)

6. answer <-[]
7. While p1!= NIL and p2!=NIL
    a. If docid[p1]==docid[p2]:
    b. append(answer,docid[p1])
    c. p1->p1.next
    d. p2->p2.next
    e. Elif docid[p1]<docid[p2]
    f. p1->p1.next
    g. Else
    h. p2->p2.next

8. Return answer

Algorithm description:

The pointers to the lists are advanced and if the docIDs are the same the docID is added to the answer list. If docIDs are not the same, the pointer pointing to the smaller docID is advanced. Once the end of either list is reached the answer list is returned.

Sample output:

```
Enter number of queries1
Enter the query with terms separated by spacehorse lion
query terms horse lion
Enter the operation sequence separated by commaAND
Number of comparisons 98
Number of documents matched 7
The retrieved documents are: [korea.s, beggars.txt, silverb.txt, aesop11.txt, aesopa10.txt, fgoose.txt, vgilante.txt, ]
```

Another example is

```
Enter number of queries1
Enter the query with terms separated by spacehorse lion donkey
query terms horse lion donkey
Enter the operation sequence separated by commaAND,AND
Number of comparisons 117
Number of documents matched 1
The retrieved documents are: [aesopa10.txt, ]
```

- X AND NOT Y

Approach:

This is equivalent to intersection for postings in X which is NOT in Y.

AND_NOT(p1,p2)

9. answer <-[]
10. While p1!= NIL and p2!=NIL
    a. If docid[p1]==docid[p2]:

b. p1->p1.next
c. p2->p2.next
d. Elif docid[p1]<docid[p2]
e. append(answer,docid[p1])
f. p1->p1.next
g. Else
h. p2->p2.next

11. Return answer

Algorithm description:

The docid's are incremented and only if doc ID of p1 is not equal to p2 in sorted order, then the posting is added. This is ensured by maintaining the pointer p1 lesser than pointer p2.

- X OR NOT Y

Approach:

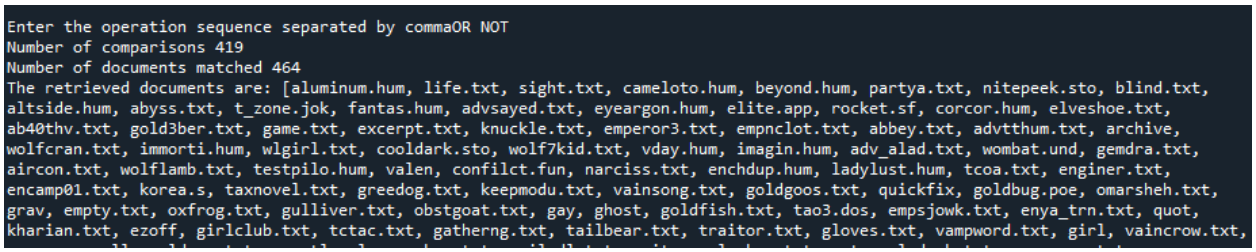
This is equivalent to Union of postings in X , NOT Y.

NOT(Y):

return [x for x in all_docids if x not in Y]

Algorithm description:

The same Union operator is applied over X and NOT Y. For finding NOT Y, we shuffle through all docid's and pick ID's not present in Y.



<u>Execution of code:</u>

- Execute intersection_and_union.py
- Enter number of queries
- For each query:
  - Enter the search string. After preprocessing, the terms to search will be shown
  - Enter appropriate number of operators to match the number of terms. If number of terms is k, then operators must be k-1
  - You will be shown the output (with document names) and prompt moves to request for next query until the number of requested queries has been processed.

Special Note:

- If an unfound term is present with AND operator, then no result is returned
- If an unfound term is present with OR operator, then that operator is as good as being skipped
- With AND NOT and an unfound term, it will match with all document ID's

WITH OR NOT and an unfound term, the entire doc ID's will be returned