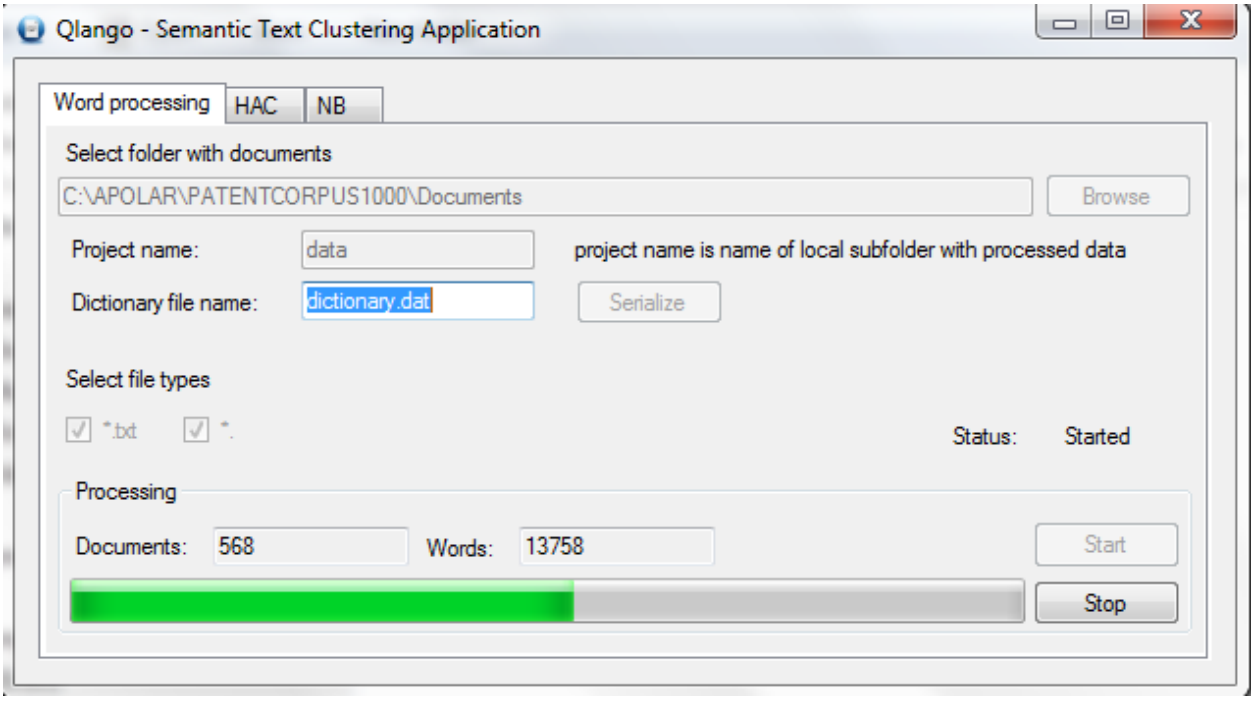
QLANGO user manual

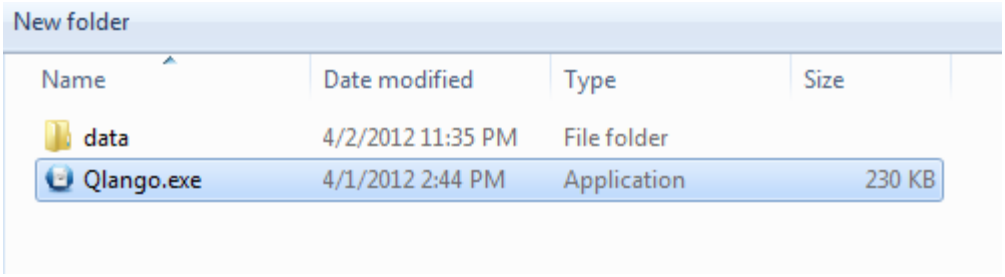
Qlango is elementary document clustering tool written by Andrew Polar. It is single executable program without libraries written in C#. It uses two algorithms Hierarchical Agglomerative Clustering (HAC) and Naïve Bayes (NB). The processing needs following steps:

1. Enumerating documents, reading each document and enumerating words and building document-term matrix.
2. Selecting training sample. This step is optional for HAC and mandatory for NB.
3. Processing document-term matrix using training sample when available.
4. For testing purposes it may estimate accuracy of clustering. For estimation of accuracy user should provide result of independent clustering for comparison.

All documents must be in ASCII format. To perform first step user needs only to navigate to top folder that contains other folders and documents and hit start button. The processing progress will be shown by progress bar along with number of documents and words.

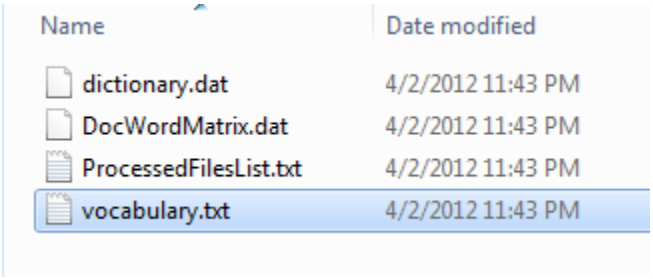


The project name is the name of subfolder with all processed data. This subfolder will be created in the current directory



.

Button 'serialize' saves the dictionary to the file. It is optional and not necessary for the further processing. The content of 'data' folder is either 2 or 4 files (when 'serialize' option is used).



Dictionary.dat is serialized dictionary class. DocWordMatrix.dat is document-term matrix in binary format. ProcessFilesList.txt is ASCII file that holds the list of processed documents in the order they were processed. Vocabulary.txt is list of words recorded after filtering stop words and stemming. This is the fragment of vocabulary.txt:

```
0 field
1 invent
2 relat
3 metal
4 cast
5 foundri
6 oper
7 particular
8 method
9 form
10 sand
11 core
12 mold
13 element
14 background
15 typic
16 compris
```

Here is the fragment of ProcessedFilesList.txt:

```
0, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6286581.txt
1, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6363995.txt
2, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6386264.txt
3, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6401798.txt
4, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6431253.txt
5, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6435259.txt
6, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6435262.txt
7, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6463991.txt
8, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6467525.txt
9, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6467529.txt
10, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6502620.txt
11, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6505670.txt
12, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6505671.txt
13, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6513567.txt
14, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6516863.txt
15, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6516864.txt
16, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6523597.txt
17, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6554049.txt
18, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6554050.txt
```

It has 3 comma separated fields. First is sequential number, second field is either known category or -1 if category is not known and third field is file name. In case user wish to provide training sample, he(she) needs to change -1 in the second field into known category like in the fragment below:

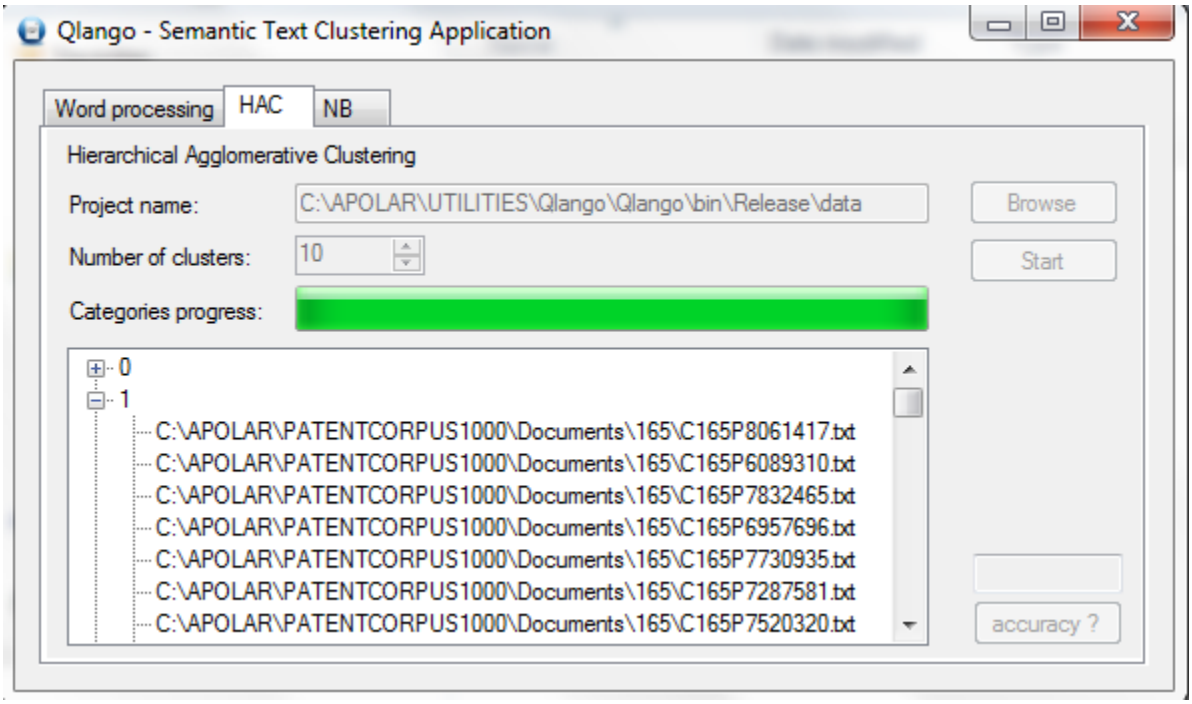
```
9, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6467529.txt
10, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6502620.txt
11, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6505670.txt
12, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6505671.txt
13, 0, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6513567.txt
14, 0, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6516863.txt
15, 0, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6516864.txt
16, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6523597.txt
17, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6554049.txt
18, 1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6554050.txt
19, 1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6564852.txt
20, 1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6591891.txt
21, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6595267.txt
22, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6615901.txt
23, -1, C:\APOLAR\PATENTCORPUS1000\Documents\164\C164P6619378.txt
```

If user knows that files 13, 14, 15 belong to the same cluster, he(she) marks them with same integer, in this case it is 0. On the same reason files 18,19,20 are marked as 1. So all other known groups must be assigned sequential integer representing index of the cluster. Files marked as -1 are to be clustered. The format of DocWordMatrix.dat is elementary. It is 10 byte fragments for every word in every file. First 4 bytes is sequential number of the file matching index in ProcessedFilesList.txt. Second 4 byte fragment is the index of the word from the dictionary matching index from vocabulary.txt and third 2 byte fragment is number of times this particular word is observed in this particular document.

After word processing is finished QLANGO can be terminated and restarted later, because other processing is independent. Also several different data corpuses can be processed and saved into different folders.

For the next step user should navigate to HAC or NB tab. HAC does not require training sample but when it is specified in ProcessedFilesList.txt it uses it. NB requires training sample, so when it is not specified for NB program throws exception. In case of NB the number of clusters specified by spinner should match the number of clusters specified in ProcessedFilesList.txt. The processing result is shown in the

picture below. The project name is actually the data folder that was created in Word processing tab. It is the folder that holds DocWordMatrix.dat and ProcessedFilesList.txt. These files are necessary for clustering, other two are not involved.



Program builds the tree for the categories and shows the file names as items in the tree nodes. The functions that compare obtained clustering with expected clustering for accuracy test are provided but not associated with any button for the moment. The output clusters to files is not provided either. Since it is open source users are expected to add these parts when necessary.

The code considered by author as elementary educational tool for students or beginners. HAC is near 500 lines, NB is near 600 lines. The code, however, is optimized for fast processing. NB, for example, needs few seconds to categorize 5000 documents into 50 clusters with training sample near 8 percent of all files and shows accuracy near 80 percent. These elementary and known for, at least, 30 years algorithms were chosen as unconditional winner after comparison with following:

1. Latent Semantic Analysis
2. Probabilistic Latent Semantic Analysis
3. Support Vector Machine
4. Latent Dirichlet Allocation
5. K-means
6. K-nearest neighbors
7. N-grams
8. Random Forest
9. Self-Organizing Maps