Title :Text Clustering With Named Entities.

Author : Nirmisha Bollampalli

1) System Description :

- It is relatively common for different people or organizations to share the same name. Given the increasing amount of information available online, this results in the ever growing possibility of finding misleading or incorrect information due to confusion caused by an ambiguous name.

- This system makes use of an unsupervised approach that resolves name ambiguity by clustering the instances of a given name into groups, each of which is associated with a distinct underlying entity.

- **Weka System Description** :(Refer ReadMe for detailed description on commands)

Step 1 : Clean the html files using nltk

Step 2 : Create arff files from the cleaned html files using TextDirectoryLoader(comes as a part of weka).

Step 3 : Load the arff files one by one into weka.

Step 4 : Apply StringToWord Filter [set options(tf idf = true,tokenisation = Word Tokeniser(unigrams)/NgramTokeniser(bi/uni+bi),stemming(Lovins Stemmerr),pruning(2.0),StopList=true and min-term frequency = 3)].

Step 5 : Apply Normalise and Discretize Filters.

Step 6 : Run Kmeans or EM (by setting no of iterations as 3000)

Step 7 : Save output from weka in arff format.

Step 8 : Create XML files for each name from the output using the provided python files and run the scorer.

-**Mallet System Description** :(Refer ReadMe for detailed description on commands)

Step 1 : Clean the html files using nltk

Step 2 : Run the following command to generate the model.(* = abby or other..)

"**bin\mallet import-dir --input path_to_Cleaned_File_Name* --output path_to_store_model\modelName.mallet --keep-sequence --remove-stopwords --keep-sequence-bigrams --gram-sizes "1,2""**

Step 3 : Run the following command on the generated model to get topic_compostion.txt

"**bin\mallet train-topics --input path_to_model --num-topics 15 --num-iterations 1000 --use-ngrams true --output-state path_to_output_folder\topic-state.gz --output-topic-keys path_to_output_folder\topic_keys.txt --output-doc-topics path_to_cleaned_name_folder\topic_compostion.txt** "

Step 4 : Create XML files for each name from the output using the provided python files and run the scorer.

1.1 ToolKits Used :

Mallet : - MALLET is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction,and other machine learning applications to text.I have used Mallet for Topic Modeling.

- Download Link : http://mallet.cs.umass.edu/download.php

WEKA :- Weka(Waikato Environment for Knowledge Analysis) has a number of machine learning algorithms and we have used KMeans,EM etc. for clustering large groups of unlabelled data.

- Download Link : http://www.cs.waikato.ac.nz/ml/weka/downloading.html

Python and Nltk : For cleaning the html files and for creating XML files.

1.2 Features :

Tf–idf :

Tf–idf, term frequency–inverse document frequency, is a numerical statistic which reflects how important a word is to a document in a collection or corpus. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

Unigram :

Unigram == word. A unigram model reduces a (structured) sentence to a(n unstructured) bag of words.

Bigrams :

-Bigram is a group of two words.Bigrams are particularly useful because some permutations of two adjacent words are much more common than others in any given document.

Stemming :

-Stemming refers to a heuristic process that chops off the ends of words and often includes the removal of derivational affixes.

-I have used Lovins Stemmer.The Lovins stemmer has 294 endings, 29 conditions and 35 transformation rules.

-Each ending is associated with one of the conditions. In the first step the longest ending is found which satisfies its associated condition, and is removed.In the second step the 35 rules are applied to transform the ending.The second step is done whether or not an ending is removed in the first step.

Tokenisation :

-Tokenization is the process of breaking up the given text into units called tokens.
The tokens may be words or number or punctuation mark. Tokenization does this task by locating word boundaries.Ending point of a word and beginning of the next word is called word boundaries.

-I have used Word Tokenizer and NGram-Tokenizer.

StopWords :

-Stop words are common words like an,the etc that cause noise in the documents.

-I have used weka's StopList and Mallet's StopList to remove stopwords.

Periodic Pruning :

-As the data set is processed in order to build the dictionary, this feature will examine the dictionary after every x% of the instances and remove words which have a count of 1. For large corpora we might run out of memory,so periodic pruning can prevent this from happening.

MinTerm Frequency :

-By setting minTerm Frequency we specify when to consider a word(For eg : minTermFReq = 2 ,states that a word should be considered only if it occurs more than twice)

Analysis On Features and Feature Sets :

➔ **Weka :**

Set 1 : Unigrams plus StopWords plus Stemmer (gives good performance and Stemmer plays a major role)

Set 2 : Unigrams plus StopWords plus Stemmer plus Periodic Pruning(Periodic pruning seems to have a vast increase in the performance)

Set 3 : Unigrams plus StopWords plus Stemmer plus Periodic Pruning plus tf idf (Performance remains the same).

Set 4 : Unigrams plus StopWords plus Stemmer plus Periodic Pruning plus tf idf plus MinTermFreq (Performance boosts).

Set 5 : Set 1 to 4 was performed again using bigrams or trigrams in place of unigrams .

The performance remained the same with bigrams(experimental results included) and decreased incase of trigrams(experimental results not included)

Set 6 : Unigrams plus bigrams plus StopWords plus Stemmer plus Periodic Pruning plus tf idf plus MinTermFreq (Good Performance).

➔ **Mallet :**

Set 1 : Unigrams plus StopWords (gives a good performace comparable-to(even greater-than) set 4 in Weka).

Set 2 : Bigrams plus StopWords (gives a good performace comparable-to(even greater-than) set 4 in Weka).

Set 3 : Unigrams plus Bigrams plus StopWords (gives a better performanc than set 4 in weka but performance slightly decreases when compared to Set 1 and 2 in mallet).

1.3 Clustering Algorithms :

WEKA :KMeans,EM.

- KMeans algorithm gave the best system(0.68) and EM gave a system slighly poor in performance(0.62).
- **What went wrong with EM??**

  -EM algorithm finds a local maxima and not the global maximum(main reason for why the score wa really low).

  -It requires an initial estimate of theta.Since multiple local maxima of the likelihood function are frequent in practice and the algorithm coverges only to a local maxima,the quality of the initial estimate can greatly influence the final result.

- **Solution** : - One of the solutions that I read about was to do a random restart.Take the output from the first run and then run it again.
- I was not able to implement this because of memory issues with the system.
- **Cause of slowness** of the EM algorithm arises when the E or M step does not admit an analytical solution.It becomes then necessary to use iterative methods for computation of the expectation or for the maximization, which can be computationally expensive.
- **What Worked with KMeans ??**

  -KMeans when applied with a smilar setup(as the one with EM) gave a slightly better performace than EM but when filters like Normalise and Discretize were applied the performance boosted to 0.68

  - KMeans worked better also because the number of clusters were provided.That is the difficult part in KMeans lies in finding the no of clusters where as EM is a bit flexible in that part.

- **What further improvements can be made??**

  -I noticed that setting a proper seed value before clustering gave upto 71 % (that is setting different seed values for different documents).

  -However it would be more appropriate if we can find a proper seed value that can be applied in a generic manner to all files.

  -Further,having POS tagging(Word_POS tag) and sentence level segmentation(while cleaning the html files) might help in boosting the performance further.

MALLET : General Topic Modelling , Ngram-Topic Modelling.

- With Mallet both General Topic Modelling , Ngram-Topic Modelling(N=2) gave the same score (0.65)
- But Ngram-Topic Modelling(N=1,2) gave a slightly lower score (0.64).This happens because we havenot done proper sentence level segmentation while cleaning the html files. So,system computes noisy bigrams while it reaches the sentence boundaries.
- Further while using unigrams with bigrams ,if we compute bigrams from the unigrams that make more sense there might be a considerable improvement in the task.

- **What further improvements can be made with mallet??**

  -With KMeans(or WEKA in general) we have made use of several features to prune and filter the text which we have not done with mallet.Implementing the same with mallet might improve the performance.

WEKA Versus MALLET which is better ??

-Both WEKA and MALLET work well for our task(where we assume that each document relates only to a single person.)

-However MALLET might prove to be better if we use it for tasks where each document will be considered to represent multiple topics.

1.4 Comparative Study on Scenarios:

Scenario I: How different features affect the same clustering algorithm?

-Unigrams,Stopwords Topic Modeling : 0.65

-Bigrams,StopWords Topic Modeling : 0.65

-Bigrams are generally considered to work better,but in my system,since I have not done sentence level segmentation,the system gets confused with the sentence boundaries and so there is noise when bigrams which leads to a score equal to the score produced by unigrams.

Scenario II: How the same features affect the two different clustering algorithms ?

-Unigrams,StopWords,tf idf,pruning,MinTermFreq KMeans : 0.68

-Unigrams,StopWords,tf idf,pruning,MinTermFreq EM :0.62

-As explained earlier since the no of clusters where provided (which is crucial to KMeans)KMeans works better than EM.EM ‘s performance can be increased by choosing proper theta values.

1.5 Experimental Results – Best System:

F-Score : For Individual Names

topic	BEP	BER	FMeasure_0.5_BEP-BER			FMeasure_0.5_P-IP			IP	P	PJ
	PM	PR									
Abby_Watkins			0.73	0.48	0.58	0.63	0.66	0.60	0.30	0.47	0.37
Cathie_Ely			1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Dan_Rhone			1.00	1.00	1.00	0.67	1.00	0.50	1.00	1.00	1.00
Jane_Hunter			0.38	0.60	0.46	0.30	0.68	0.19	0.09	0.20	0.55
Michael_Howard			0.47	0.39	0.43	0.51	0.50	0.52	0.07	0.13	0.71
Thomas_Baker			0.72	0.66	0.69	0.61	0.77	0.51	0.04	0.08	0.94
Tim_Whisler			0.61	0.53	0.57	0.55	0.60	0.52	0.11	0.21	0.69
Average			0.70	0.67	0.68	0.61	0.74	0.55	0.37	0.44	0.75

Overall System Run:

WePS 2007 Evaluation Package (http://nlp.uned.es/weps)

Key clustering files path: webps\truth_files

Answer clustering files path: test_files

Output evaluation files path: Evals

Measures: [P, IP, FMeasure_0.5_P-IP, BEP, BER, FMeasure_0.5_BEP-BER, PM, PJ, PR,]

Baselines: [COMBINED_BASELINE, ONE_IN_ONE_BASELINE, ALL_IN_ONE_BASELINE]

Overwrite: false

* Loading team evaluation from: Evals\mallet.eval

* Loading team evaluation from: Evals\mallet_bi_grams.eval

* Loading team evaluation from: Evals\mallet_pre_processed.eval

* Loading team evaluation from: Evals\mallet_uni+bi_grams.eval

* Loading team evaluation from: Evals\Weka_EM.eval

* Loading team evaluation from: Evals\Weka_Kmeans_Bi_filtered.eval

* Loading team evaluation from: Evals\Weka_Kmeans_Bi_filtered_post_processed.eval

* Loading team evaluation from: Evals\Weka_Kmeans_uni.eval

* Loading team evaluation from: Evals\Weka_KMeans_uni_filtered.eval

* Loading team evaluation from: Evals\COMBINED_BASELINE.eval

* Loading team evaluation from: Evals\ONE_IN_ONE_BASELINE.eval

* Loading team evaluation from: Evals\ALL_IN_ONE_BASELINE.eval

topic	BEP PM	BER PR	FMeasure_0.5_BEP-BER			FMeasure_0.5_P-IP			IP	P	PJ
ALL_IN_ONE_BASELINE	0,41	0,99	0,52	0,65	1,0	0,53	0,33	0,62	0,33		
COMBINED_BASELINE	0,33	1,0	0,45	0,86	1,0	0,77	0,33	0,62	0,33		
mallet	0,73	0,66	0,65	0,57	0,72	0,57	0,29	0,54	0,68		
mallet_bi_grams	0,69	0,67	0,65	0,57	0,73	0,54	0,33	0,59	0,68		
mallet_pre_processed	0,74	0,65	0,65	0,6	0,72	0,61	0,29	0,54	0,68		
mallet_uni+bi_grams	0,69	0,66	0,64	0,57	0,72	0,55	0,27	0,52	0,67		
ONE_IN_ONE_BASELINE	1,0	0,48	0,61	0,62	0,49	1,0	0,14	1,0	0,67		
Weka_EM	0,62	0,68	0,62	0,56	0,76	0,5	0,23	0,45	0,59		
Weka_Kmeans_Uni_Bi_filtered	**0,69**	**0,69**	**0,68**	**0,62**	**0,77**	**0,55**	**0,39**	**0,46**	**0,75**		
Weka_Kmeans_Uni_Bi_filtered_post_processed	0,39	0,46	0,76	0,71	0,67	0,69	0,65	0,76	0,6		
Weka_Kmeans_uni	0,64	0,67	0,64	0,6	0,76	0,53	0,38	0,45	0,72		

Weka_KMeans_uni_filtered	0,7	0,67	0,68	0,61	0,74	0,55	0,37	0,44	0,75

1.6 Error Analysis:

-While creating the truth files human annotators have classified some files,which there were not able to classify under any other cluster as discarded files.This resulted in system score deviation.

-Also originally(in the pdf provided to us) the empty files were asked to not to be a part of the clust.xml.But,later we were asked not to discard those files.This also resulted in system score deviation.

-As we can observe from the experimental results.By classifying irrelevant files as discarded files,the system performs better.(with WEKA)

Conclusion : We have built a Text Clustering system that is 68% efficient in classifying different topics that a set of documents(that refer to the same person) belong to.As discussed in the report there is a scope for lot of improvements to the implemented system.