Election Political Online Sentiment

PRESENTER: DENNIS HSU

CS 262

NOVEMBER 30, 2016

Outline

- Introduction
- Background
- Current Solution
- Randomization Techniques
- Implementation
- Experiment Results
- Conclusion/Future Work

Introduction

- Problem: Polls showing support for election political candidates are always changing and rarely accurate
- Polls are based on <u>voluntary</u> surveys
- <u>Social Media</u>, such as Twitter, provides an excellent source for data mining
  - Current Sentiment for each candidate
  - Not just voluntary
- Goal: To use Twitter to get the current online sentiment for political candidates (Clinton, Trump)

Introduction

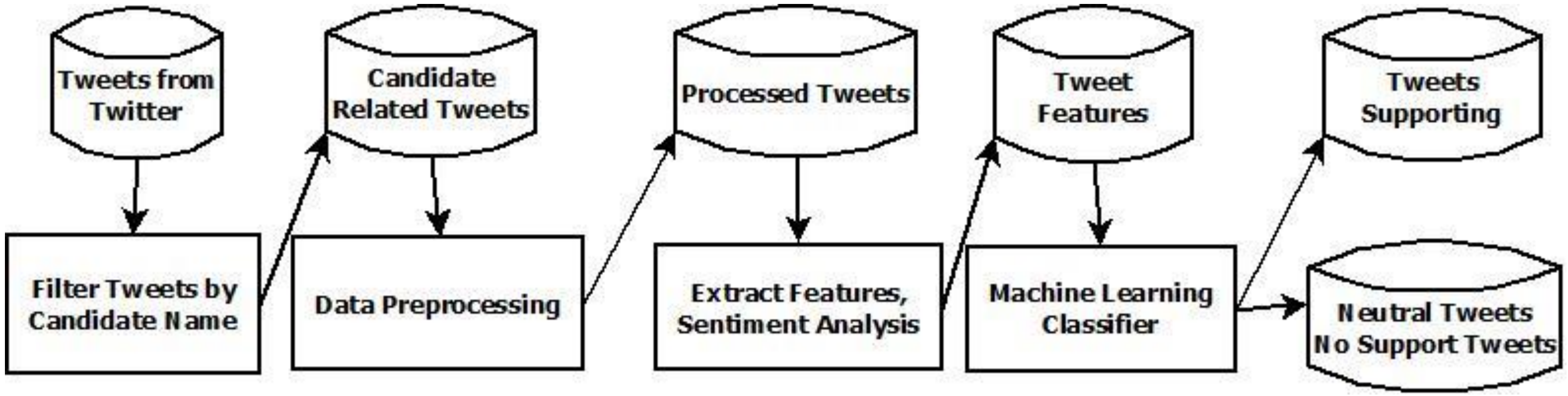
- Limitations
- Online Sentiment only
  - Vocal/Active Twitter users
  - “Echo Chamber” problem
- Online does not represent the silent majority
- Contribution: applied randomization to decrease amount of tweets stored to represent the current online sentiment/support for each political candidate

Background

- Techniques, Tools used
- Sentiment Analysis—used to label tweets as positive or negative
- Natural Language Processing—convert text/tweets into features
- Machine Learning—used to classify whether each tweet supports or not supports the candidate

Previous Solution

- Pipeline(Before)



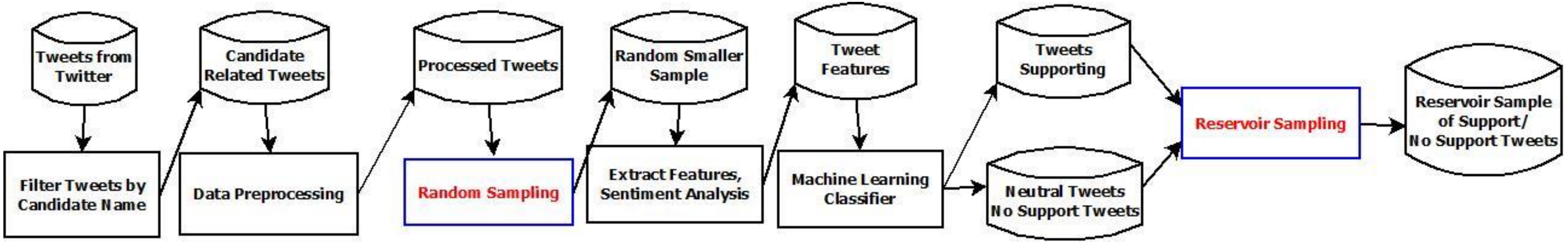
Current Solution

- Steps

1. Filter Tweets related to the candidate –(contains “Hillary Clinton” or “Donald Trump”)
2. Preprocess each Tweet into features—Sentiment Analysis, N-grams
3. Train Classifier separately for each candidate with labeled data (1 for support, 0 for oppose)
4. Use Train Classifier on an incoming stream of candidate-related tweets
5. Get current online sentiment for the candidate

Randomization Applied

- Pipeline(Randomization)



Randomization Techniques

- Randomization

1. Filter Tweets related to the candidate –(contains “Hillary Clinton” or “Donald Trump”)
2. Preprocess each Tweet into features—Sentiment Analysis, N-grams
3. Train Classifier separately for each candidate with labeled data (1 for support, 0 for oppose)
    a) **Random smaller sample used instead of a larger set**
4. Use Train Classifier on an incoming stream of candidate-related tweets
5. Get current online sentiment for the candidate
    a) **Reservoir Sampling—keep a smaller sample of tweets to represent larger set**

Randomization Techniques

- **Random smaller sample used instead of a larger set**
  - Decrease time to train classifier
  - Goal: Maintain accuracy of classifier
  - Expectation: Drop in accuracy of classifier
- **Reservoir Sampling—keep a smaller sample of tweets to represent larger set**
  - Online: Smaller Sample to keep to represent current set of classified tweets
  - Goal: Reduce amount of tweets needed to be kept
  - Expectation: Reservoir Sample should closely represent all tweets classified

Reservoir Sampling

- Keep the first tweet in memory
- When the *i*-th tweet arrives (for *i*>1):
- with probability 1/*i*, keep the new item(discard an old tweet)
- With probability 1 – 1/*i*, keep old items(ignore new tweet)
- Induction:
  - when there is only one item, it is kept with probability 1
  - when there are 2 items, each of them is kept with probability ½
  - when there are 3 items, the third item is kept with probability 1/3, and each of the previous 2 items is also kept with probability (1/2)(1-1/3) = (1/2)(2/3) = 1/3
  - by induction, it is easy to prove that when there are *n* items, each item is kept with probability 1/*n*
- From: https://en.wikipedia.org/wiki/Reservoir_sampling

```
(*
  S has items to sample, R will contain the result
 *)
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i = k+1 to n
    j := random(1, i)   // important: inclusive range
    if j <= k
        R[j] := S[i]
```

Implementation

- My Laptop was stolen Dec.5th evening, lost data/results/powerpoint; had to redo it from scratch; All of the following slides were from the before results(will have update slides later)
- Tweepy—Filtered tweets related to “Hillary Clinton” and “Donald Trump”
  - 1,000,000 tweets per candidate filtered on election day (Nov 8, 2016) and day before (Nov 7)
- Dataset—“Hillary Clinton”
  - Labeled tweets for training
  - 402 tweets used to train classifier
  - Smaller random sample: 200 tweets used to train classifier
  - Balanced dataset: half positive, half negative/neutral

Implementation

- Tweet Pre-Processing to reduce noise, spam—Regex, Natural Language ToolKit(NLTK)
- Filter out tweets with links; filter out retweets
- Remove similar tweets
  - Similarity Index: 0.6 was found to be the best
  - Used Python’s diff library SequenceMatcher
- Remove symbols, punctuation
- Tokenization into words
- Lemmatization—Base form of words to reduce noise

Implementation--Example

- Example filter
- Removed
  - RT @joshtpm: Hillary Clinton's popular vote lead now stands at 2.654 million votes, a 2 percentage point lead over Donald Trump, 48.2% to 4…
  - "CPAC 2013: Donald Trump: Immigration reform is a ‘suicide mission’ for GOP" http://t.co/WdMLJcXZLL by @SethMcLaughlin1
- Kept
  - 'A lot depends on who the real Donald Trump is.' - @BT_SDSC @PerthUSAsia #perthusasiatalks
  - a lot depends on who the real Donald Trump is perthusasiatalks

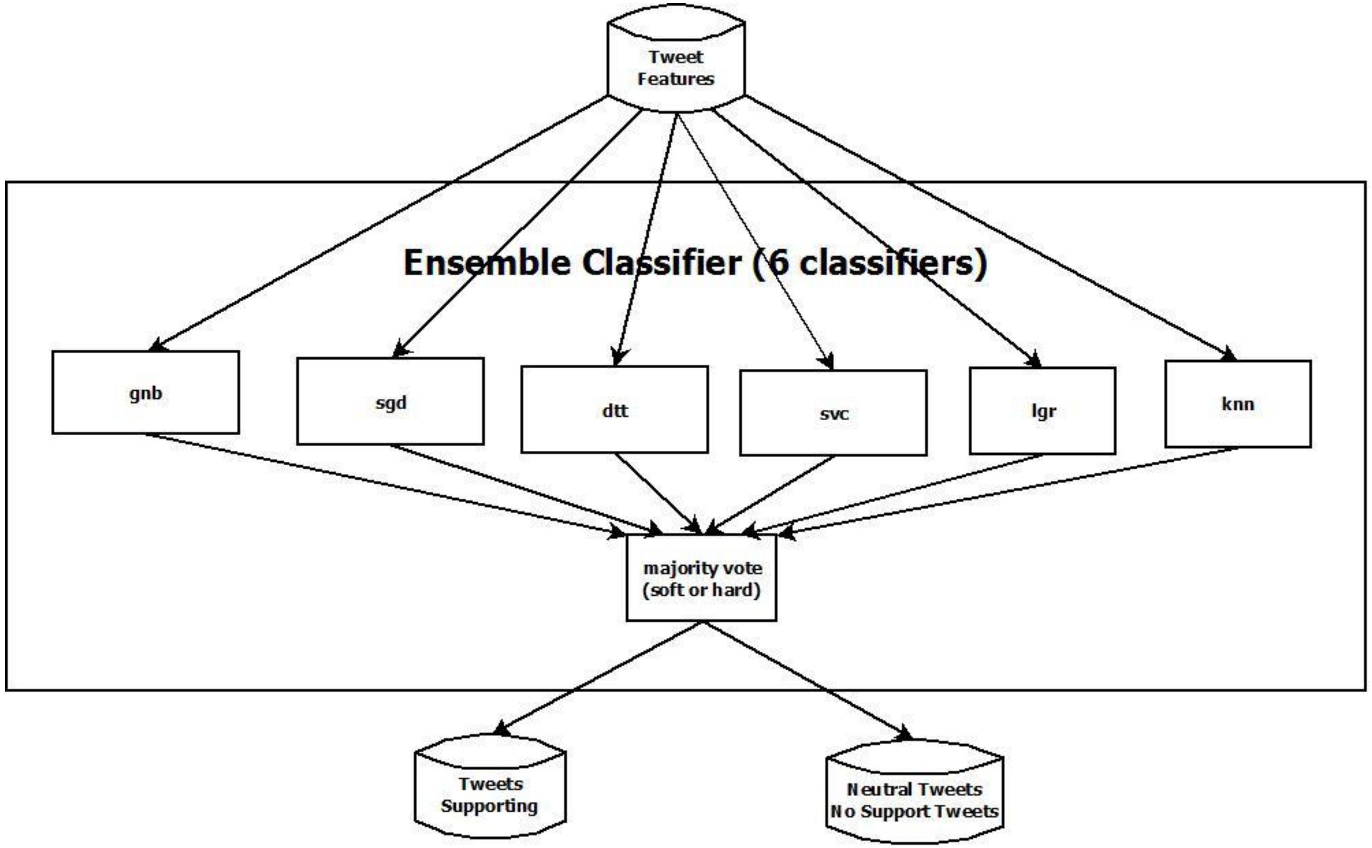
Implementation—Get Features

- Sentiment Analysis—lexicons—get a sentiment score for each tweet
  - Sentiwordnet—combine positive/negative scores
  - AFINN—get sum of scores
- N-grams—convert text into set of n-grams as features
  - Unigrams, bigrams—convert into sets of one word/two words as well as one character/two characters
- Combine together into set of features
- Convert features into a feature matrix using a vectorizer

Implementation--Classifier

- Train Classifier using feature matrix generated from labeled tweets
- Classifiers used
  - Gaussian Naïve Bayes(gnb)
  - Support Vector Classifier(svc)
  - Logistic Regression(lgr)
  - k-Nearest Neighbors(knn)
  - Decision Tree(dtc)
  - Stochastic Gradient Descent(sgd)
  - Random Forest Classifier(rfc)
  - Ensemble Classifier(gNB,SVC,LGR,kNN,DT,SGD)—(soft voc, hard voc)
    - Hard: majority vote
    - Soft: weighted probabilities of the vote
- Cross Validation(5-fold)—split dataset into 5 and trained/tested 5 times; collected f1 score

Ensemble Classifier



Implementation—Resevoir Sample

- Size of all tweets: 10,000
- Size of Reservoir Sample: 1,000
- Based on the algorithm in one of the previous slides, implementation was done
  - Loaded first 1000 tweets (if count < 1000)
  - Picked random number
  - If random number is less than or equal to 999, do replacement

```
r = random.randint(0, count)
if r < 999:
    #do prediction and replacement
```

- Compared with taking a random sample of 1000 from all tweets so far each time a new tweet comes in

Experiment Results—Old

- **<u>Stolen Laptop: Lost dataset,results, graphs, tweets of Hillary Clinton and Donald Trump from Election Days</u>**
- Data below was from what I remembered and presentation
- Testing of smaller sample used to classify: Hillary Clinton
  - 402 tweets: best classifier was Ensemble Classifier(~71-72%)
  - 200 tweets: best classifier was SVC(~71%)
- Testing of smaller sample kept: Hillary Clinton
  - From what I remember, Hillary had an approval rating of around 42% with full 10,000 tweets
  - Random Smaller Sample of 1000: approval rating varied above and below the full set, but error average was 1.8%
  - Reservoir Sampling of 1000: approval rating was below the full 10,000 tweets, but only by average error rate of 1.2%

Experiment Results—New(After Presentation)

- (**Update: Since laptop was stolen, I have re-mined tweets again this time focusing on Donald Trump (Tweets from Dec. 6th) and his current support**
- Labeled sample of 400 tweets (200 positive, 200 negative)
- Compare With smaller random sample of 200 tweets for training
- For reservoir sample testing: used 6000 tweets this time around of Donald Trump from Dec. 6, 2016
- Tested accuracy comparison of averages for 400 tweet-classifier on total, reservoir, and random sampling

Experiment Results—400 Sample Size; (f1 scores for each classifier)

count(word+score)(2-3) **93.146s**	tfidf(char+score)(1-4) **102.172s**	tfidf(word+score)(1-3) **107.248s**
svc: 0.649561568085	svc: 0.689612222526	**svc: 0.684670519728**
gnb: 0.5208608176	gnb: 0.601457339317	gnb: 0.559498762542
lgr: 0.641805527404	lgr: 0.694846586696	lgr: 0.679301918208
sgd: 0.632294487482	sgd: 0.579201178901	sgd: 0.584666727607
knn: 0.466596353505	knn: 0.596530272476	knn: 0.600343549055
dtc: 0.571007872994	dtc: 0.583951333026	dtc: 0.528953643075
rfc: 0.463463336887	rfc: 0.575218866183	rfc: 0.533389033991
soft voc: 0.594174633452	soft voc: 0.674683097813	soft voc: 0.60144131876
hard voc: 0.608975062333	**hard voc: 0.696358144974**	hard voc: 0.679381774203

Experiment Results—200 sample size

tfidf(word+score)(1-3) **15.493s**	tfidf(char+score)(1-4) **21.516s**	tfidf(char_wb+score)(1-4) **13.687s**
svc: 0.519064566296	svc: 0.569200893497	svc: 0.555067332602
gnb: 0.59594614386	**gnb: 0.635151969981**	gnb: 0.60772945497
lgr: 0.520354460224	lgr: 0.577686740766	lgr: 0.540404076944
sgd: 0.642713979644	sgd: 0.536354228685	sgd: 0.548920894017
knn: 0.541998670405	knn: 0.546332346768	knn: 0.543829672625
dtc: 0.472733477694	dtc: 0.587154029445	dtc: 0.566337112722
rfc: 0.4963818842	rfc: 0.491128224627	rfc: 0.598614631165
soft voc: 0.55935552889	soft voc: 0.6133339599	**soft voc: 0.645151324792**
hard voc: 0.573636373611	hard voc: 0.613755190686	hard voc: 0.598973556753

Experiment Results—Reservoir

6000 Tweets(Total) Run time: 102.007 seconds

1000 Tweet Random Sampling runtime: 118.039 seconds

1000 Tweet Reservoir Sampling runtime:**54.332 seconds**

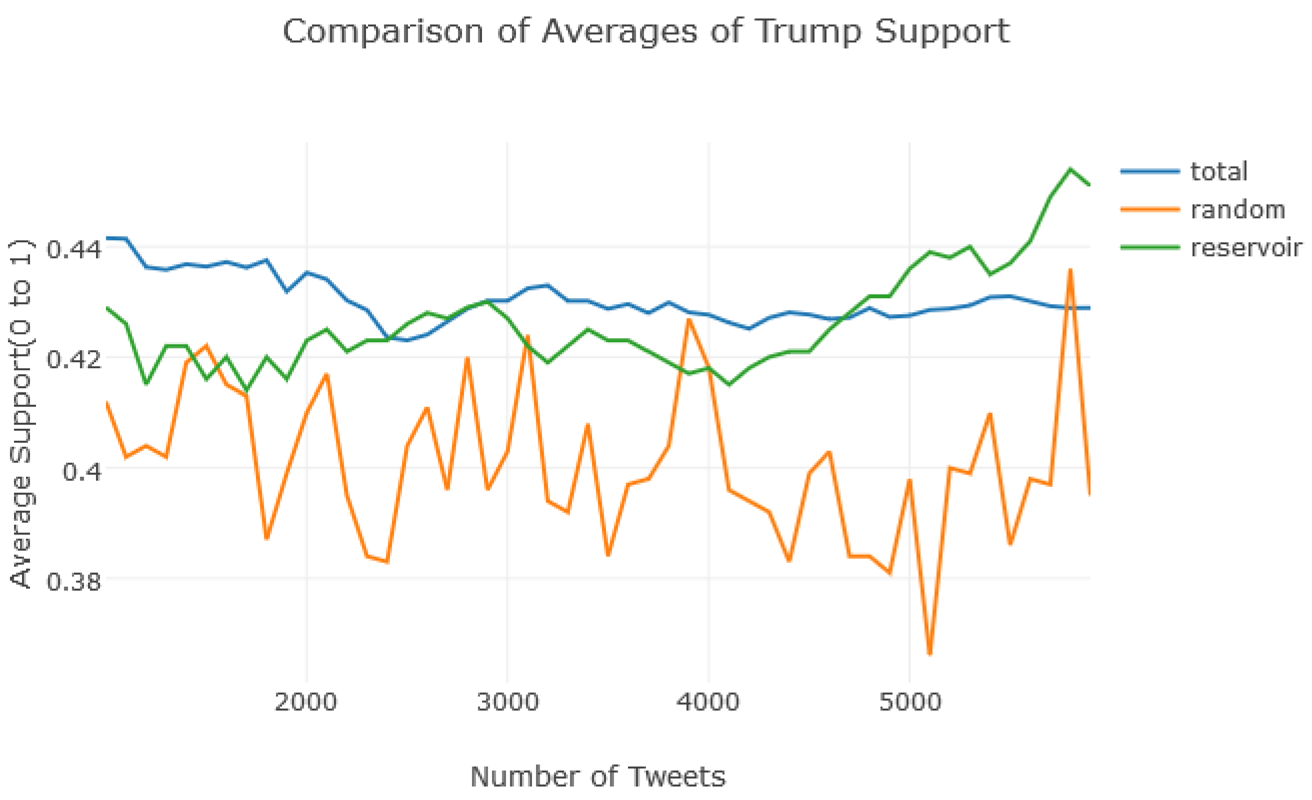
Experiment Results—Reservoir

Total versus Reservoir:

0.044179291646% avg. error

Total versus Random:

0.067820708354% avg. error



Conclusion & Future Work

- Have tested two random sampling techniques to improve time of pipeline
  - Used random smaller sample to train classifier to decrease time taken while maintaining accuracy—found out larger sample is better than smaller sample (obviously)
  - Used reservoir sampling to decrease amount of tweets needed to be kept (save space) as well as save time compared to taking a random sample each time
- Can be applied to other social media; pull dataset from Yelp, Facebook, etc.
- Can be used in other political elections or even on approval ratings for bills/propositions
- Future work
  - Apply to Donald Trump Tweets over more time
  - Apply to another elected official/candidate
  - Change classifier into 3 classes: positive, neutral, negative and retest pipeline and randomization

References

Fan Yu. Melody Moh. Teng-Sheng Moh. "Towards Extracting Drug-Effect Relation from Twitter: A Supervised Learning Approach." IEEE International Conference on Intelligent Data and Security to be held New York. April 2016. Accessed Nov 2016.

Liang Wu (MSCS), Teng-Sheng Moh and Natalia Khuri, "Twitter Opinion Mining for Adverse Drug Reactions," Proceedings of the 2015 IEEE International Conference on Big Data (BigData), Santa Clara, California, Oct. 2015, pp.1570-1574. Accessed Nov 2016. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7363922

Tweepy. Last Retrieved on Nov 2016. http://www.tweepy.org

"Reservoir Sampling". Wikipedia. Last Retrieved Dec 2016. https://en.wikipedia.org/wiki/Reservoir_sampling

Stefano Baccianella et. al. "SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining". Istituto di Scienza e Tecnologie dell'Informazione. Last Retrieved Nov 2016. http://nmis.isti.cnr.it/sebastiani/Publications/LREC10.pdf , http://sentiwordnet.isti.cnr.it/

Finn Årup Nielsen. "AFINN". Informatics and Mathematical Modelling, Technical University of Denmark. Last Retrieved Dec 2016. http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

W. B. Cavnar, and J. M. Trenkle, "N-Gram-Based Text Categorization", in Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, pp. 161-175, 1994. Accessed Jun 2016. http://odur.let.rug.nl/~vannoord/TextCat/textcat.pdf

Code: https://github.com/denniseh7/CS262