**Sentiment Analysis**

In this assignment we are using natural language processing to detect the sentiment of text. We will work with a set of real tweets collected by researchers who developed one of the first approaches to sentiment analysis of tweets and build our own tweet sentiment analyser.

**Data**

In the Training Tweets datasets contain following variables:

* The gold polarity of the tweet (0 = negative, 2 = neutral, 4 = positive, = not given)
* The id of the tweet (2087)
* The date of the tweet (Sat May 16 23:58:44 UTC 2009)
* the query (lyx)
* the user that tweeted (e.g. robotickilldozr)
* the text of the tweet (e.g. Lyx is cool)

In Basic Sentiment words:

* There’s a widely used subjectivity and sentiment lexicon that I’ve extracted data from. Each line consists of a word, followed by the typical sentiment of that word without any additional context, indicated by the string positive or negative

Fine-grained Sentiment Words

* The full lexicon from above also includes information about the strength of the sentiment: “weaksubj” indicates weak sentiment, and “strongsubj” indicates strong.

Inverse Index

* There are some inverse indexes that, for each word, give the IDs of tweets that contain that word.

**Approach**

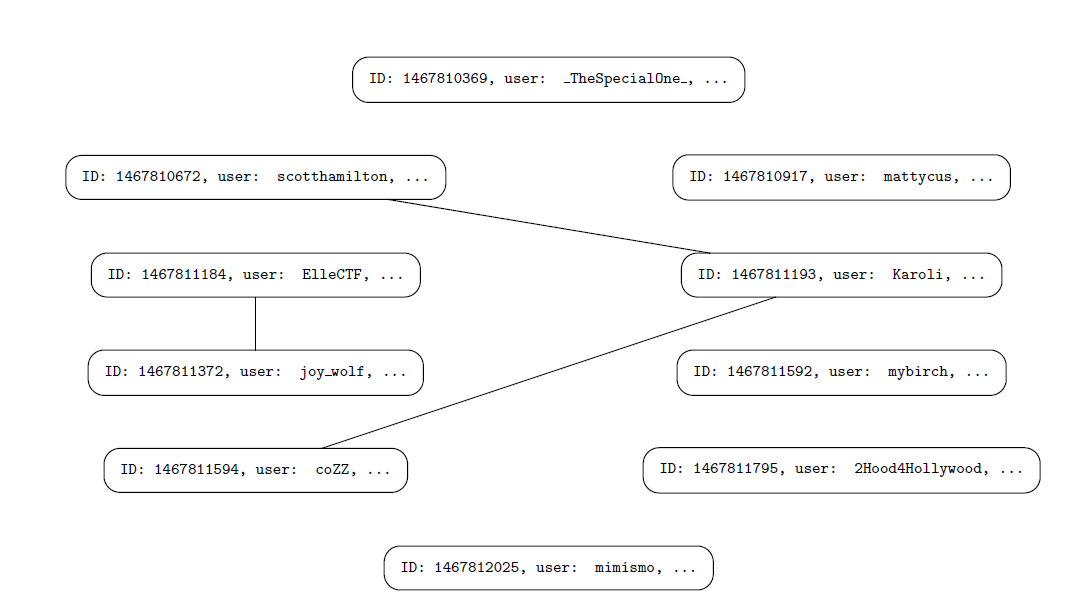
We will be adding attributes and methods to existing classes given in the code bundle accompanying these specs. Where it’s given, we will use exactly the method stub provided for implementing my tasks.

The two classes provided are Tweet and TweetCollection. The former represents an individual tweet, and the latter a collection of them. Note that the Tweet class contains two enumerated types: Polarity represents the possible sentiment polarity values for a tweet (POSitive, NEGative, NEUTral or NONE); and Strength, for the strength of polarity (WEAK, STRONG),

Tasks:

1. Using Basic Sentiment words to make predictions on individual tweets’ polarity.
2. Build a graph that enhanced predictions from basic sentiment, by creates a graph structure connects similar sentiments tweets.

Example:



**Approach Task 1:**

We implemented a simple key based method for sentiment analysis of tweets, counting up the numbers of positive and negative words in a tweet to determine the predicted polarity of the tweet. (This differs from the gold polarity, which is what has been decided as the true

polarity of the tweet; the goal is to see how well we can predict it based on the content of the tweet)

Implementation includes:

* Tweet constructor
* Tweet\_Collection constructor
* Getter and Setter functions
* PredictedPolarity () function
* importBasicSentimentWordsFromFile()
* getBasicSentimentWordPolarity
* Accuracy ()
* Coverage ()

Rules when assigning Sentiment:

* If there are no positive or negative words in the tweet, assign predicted sentiment NONE.
* If there are more positive than negative words, assign predicted sentiment POS.
* If there are more negative than positive words, assign predicted sentiment NEG.
* Otherwise, assign NEUT.

Accuracy Definitions:

* Count up the number of tweets for which the predicted polarity is the same as the gold polarity, as long as this is not NONE (numCorrect).
* Count up the number of tweets for which a prediction is made, i.e. not NONE (numPredicted).
* Accuracy is the proportion numCorrect / numPredicted.
* Don't include tweets in either numerator or denominator that have gold polarity NONE.

Coverage Definitions:

* Count the number of tweets for which a prediction is made, that is not NONE
* Coverage is the proportion of : number of predicted tweets / total number of tweets
* Don't include tweets in either numerator or denominator that have gold polarity NONE.

Other Notes:

* That the matching with words in the sentiment words file should be done using the tokenisation provided by getWords().

**Approach Task 2:**

From results of Task 1 we will notice that we ended up with quite a few unlabelled tweets, because several of them didn't contain words from the sentiment lexicon. Therefore we will build a graph that links together `similar' tweets (for some definitions of similarity). Then we will propagate sentiment labels via the edges in the graph, by identifying connected components in the graph and the majority labels in those connected components.

graph-like properties,

Implementation includes:

* AddNeighbour()
* numNeighbour()
* deletAllNeighbour()
* getNeighbourTweetIDs()
* isNeighbour()
* importInverseIndexFromFile()
* constructSharedWordGraph()
* annotateConnectedComponents()
* numConnectedComponents()
* propagateLabelAcrossComponent()
* propagateMajorityLabelAcrossComponents()

Rules when propagating sentiment:

* If there are no positive or negative tweets in the component, majority sentiment is NONE.
* If there are more positive than negative tweets, majority sentiment is POS.
* If there are more negative than positive tweets, majority sentiment is NEG.
* Otherwise, NEUT.

Majority label definition:

* Majority label is defined as whichever of POS or NEG has the larger count;
* if POS and NEG are both zero, majority label is NONE
* otherwise, majority label is NEUT
* If keepPred is True, only tweets with predicted label None are labelled in this way
* otherwise, all tweets in the component are labelled in this way