Tell me a good joke !

Evelina Dumitrescu, SCPD

April 23, 2015

Contents

- Project description
- Current state of the project
- Further work

Project description

- We want to determine what makes a good joke based on users' comments
- positive/negative comments
- Corpus based on comments from joke pages available on Facebook(ie: 9GAG, Joke Of The Day)

Current state of the project

- Building the corpus
- Preprocessing the text
- TF-IDF computation

Current state of the project (2)

Building the corpus

- used Facebook Graph API
- built a custom crawler to extract posts and comments from public pages with jokes(EnglishJokes, FunnyOrDie, JokeOfTheDay)
- maximum 200 posts/page and 200 comments/post
- classified the comments into positive/negative/spam

Preprocessing the text

- removed all unicode characters(Arabic/Chinese language, other signs)
- substituted emoticons with labels (eg: ":)" happyface)
- tokenized sentences with NLTK[1], removed punctuation signs and stopwords
- replaced abbreviations (e.g: won't => will not, ain't => is not)

[1] http://www.nltk.org/

Preprocessing the text(2)

- removed repeating letters and sequence of letters
  - e.g: liiiiiiike => like; hahaha => haha lolololol => lol
  - removed the letters until the word is found in WordNet[2]dictionary
- spellchecking - used Python Enchant[3]wrapper
  - we consider the dictionary's suggestion only if the edit distance is <= 2, otherwise we return the word
- added some abreviations from Netlingo[4]to Enchant's dictionary (e.g: wtf, lol, lmao)
- stored the lemmatized form of the word (e.g.: laughing => laugh, crying => cry)

[1]https://wordnet.princeton.edu/
[1]http://pythonhosted.org//pyenchant/
[1]http://netlingo.com/

TF-IDF computation

- used Scikit-learn[5]
- TfidfVectorizer for computing the TF-IDF score
  - unigrams/bigrams/trigrams
  - L2 normalized
  - sublinear TF(replaces tf with 1 + log(tf))
- SelectKBest with chi square for selecting the most K relevant features

[1] scikit-learn.org/

TF IDF computation(2)

Feature	KBest+	KBest-	Polarity+	Polarity-
aha lol aha	0.042507	0.0	0.99103	0.008969
hilarious	0.037383	0.0	0.997942	0.002057
joke lol	0.046482	0.0	0.200838	0.799162
bastard	0.0	0.045490	5.12533e-10	1
brilliant	0.037383	0.0	0.98534	0.014660
crazy love	0.037383	0.0	0.999005	0.000994
aha lmao	0.046482	0.0	0.5	0.5
horrible	0.0	0.041849	1.41411e-07	1
horrific	0.0	0.031722	0.0233521	0.976648
break heart	0.0	0.033744	0.778643	0.221357
disgusting	0.0	0.043581	1.69705e-07	1
yes :)	0.030160	0.0	0.375703	0.624297

Table: Out results compared to lexicon polarities from UClassify [6]

[1] http://uclassify.com/

Further work(1)

Classify new comments

- Multinomial Naive Bayes (MultinomialNB in Scikit-learn)
- In order to classify new comments, take into account also the TF-IDF score, not just the probabilities $\rightarrow$ how frequent occurs the term in a specific class, but also how relevant it is
- "Transformed Weight-normalized Complement Naive Bayes (TWCNB)"

Accuracy of our solution

- K-fold cross validation (StratifiedKFold in scikit-learn)
- Use testing samples for comments and see if classifier's prediction corresponds to our judgement $\rightarrow$ identify false positive (FP), false negative (FN), true positive (TP) and true negative (TN) cases
- Compute F1, precision (positive predictive value), recall (sensitivity, true positive rate) scores for measuring the efficiency
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1 = 2 * \frac{Precision*Recall}{Precision+Recall}$

Questions

?