Intelligent Text Analyzer and Rating Identifier with Visualization

Prasanna Kumar Rajendran
College of Computing and Informatics,
University of North Carolina, Charlotte
prajend1@uncc.edu

Senthil Kumar Karthikeyan
Department of Data Science and Business Analytics,
University of North Carolina, Charlotte
skarthik@uncc.edu

Sankalp Gabbita
Department of Data Science and Business Analytics,
University of North Carolina, Charlotte
sgabbita@uncc.edu

Abstract

This paper is based on the idea of integrating the building block of data analysis and predictive systems- Natural Language Processing, Artificial Intelligence, and Visual Analytics. The inspiration behind this paper was the online Question and Answer based site – Quora. The users are tending to put in a lot of suggestion for a question put forth by a user. It will be time-consuming for the user to read all the comments and come to a final conclusion. This paper deals with a problem-solving strategy to visually explain all the comments consolidated together. The natural language processing of the text is a key for the final visualization. The text is read for overall positivity, negativity, and neutrality. In a way to extend this paper, we are even trying to implement this to star rating system. Where the rating for anything says for movies, products are given based on the comments what is put in by the customers on the movie or product, anything which is under investigation of being rated. The blind 5 Star rating on just clicking on them is to be preceded by this new system suggested.

1 Introduction

This paper describes the design and development of Text Analyzing and Visualization technique using Natural Language Toolkit (NLTK). NLTK is a python based pipeline framework, which provides most of the common core natural language processing (NLP) steps, from tokenization to summary resolution. We describe the Data Extraction logic (section 2), the original design of the system and its strengths (section 3), the set of provided Inputs and how system process them (section 4), and Visualization techniques for Processed text (section 5). Even there are few text processing techniques, the one discussed here in this paper is unique and first ever thought of its kind.

2 The Data Extraction

Quora: Quora is a question-and-answer website where questions are asked, answered, edited and organized by its community of users. The firm was founded in June 2009, and the website was made available to the public on June 21, 2010.Users can collaborate by editing questions and suggesting edits to other users' answers. Given a topic and question is posted related to a topic, we can see the answers to the topic, a number of up votes and down votes to an answer, comments to a particular answer and related questions on the same topic can be seen on the particular web page. By using web scraping the contents of a particular question in the web page can be extracted. The extracted data can be stored in a database. A relational schema approach can be used to store the data in the database.

Web scraping: Web scraping is a computer software technique of extracting information from websites. This is accomplished by either directly implementing the Hypertext Transfer Protocol (on which the Web is based) or embedding a web browser. Web scraping is closely related to web indexing, which indexes information on the web using a bot or web crawler and is a universal technique adopted by most search engines. In contrast, web scraping focuses more on the transformation of unstructured data on the web, typically in HTML format, into structured data that can be stored and analyzed in a central local database or spreadsheet. Web scraping is also related to web automation, which simulates human browsing using computer software. Uses of web scraping include online price comparison, contact scraping, weather data monitoring, website change detection, research, web mashup and web data integration.

This can be accomplished using Python programming by implementing a module called Beautiful Soup. Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

url = 'https://www.quora.com/What-is-computer-programming'
html = urlopen(url).read()


soup = BeautifulSoup(html, "html.parser")

# Retrieve all of the anchor tags
tags = soup('p')
for tag in tags:
    # Look at the parts of a tag
    print('Contents:', tag.contents[0])
```

Figure1:*Sample Web scraping from Quora through Python*

with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

```
C:\Python34\python.exe C:/Users/sntil/PycharmProjects/new/si.py
Contents: [<strong class="ask_interstitial_title" id="__w2_ojH5tTA_interstitial_ti
Contents: ['What are the best programming languages to learn today?']
Contents: ['Are algorithms computer science or computer programming?']
Contents: ['What are the best programming languages to learn today?']
```

Figure2: *Web Scraping output from Quora through* ***BeautifulSoup***

The output for the above code was related questions for a given question. In this way the answers to the question and related contents could be extracted and stored directly in the database.

3 System Design and Development

Our system designed for text analyzer looks like below. Previously, when combining multiple natural language analysis components, each with their own ad-hoc APIs, we had tied them together with custom glue code. The core processing unit, which fetches the extracted data from the Data Base and fed as Input to the NLP unit. The initial version of the system is shown below. The text is processed using the below AI methods to get the overall idea about the answers. The various Artificial Intelligence algorithms used for the data processing and text mining are discussed below. The huge and long texts are broken down to single key texts using these algorithms and processed for ranking priority.

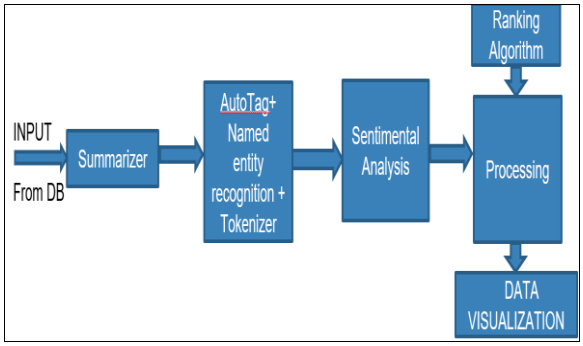

Figure3: *Overall architecture of the Text Analyzer system. The data extracted from the Quora system is preprocessed and sent to the various blocks of text processing algorithms.*

- **Summarizer**- It is an advanced AI algorithm for summarizing the text with the option of generating context-controlled summaries. This algorithm takes in huge blocks of unstructured text, and extracts core topic and sentences based on a number of times the topic is touched upon. This is an Artificial Intelligence way of processing precise idea from a large text. In short, this technique uses Lexical analysis, Syntactic Analysis and Pragmatic Analysis of the given text.
- **AutoTag**- Extracting the important keywords from the text. This is the process of filtering out the core key words that the answer to any question in Quora would try imply on. This uses the processing algorithms like Chunking and Stemming. Chunks can be represented like tags or like parse trees.

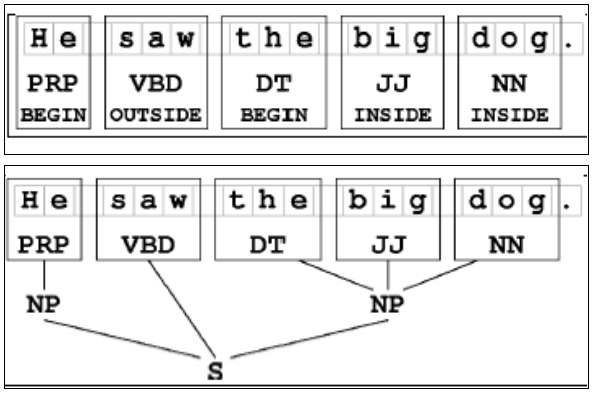

Figure4: *Processing of text using the chunk parser.*

- **Named Entity Recognition(NER)**- The segregated keywords after AutoTag process need to be identified based on the proper nouns. This algorithm works as below, All the independent evidence exists for one type of NE, this method can be directly applied to classify such Named Entity. We demonstrate how to apply our method to classify three common type of Named Entities:

person (PER), organization (ORG), and location (LOC) using the Conditional Random Fields. A non-NE is annotated as O. The Conditional Random Fields for NER can formulated like below the extracted formulation from *"A Simple Semi-Supervised Algorithm for Named Entity Recognition*"

The conditional probability of a sequence of labels given the corresponding input sequence. Let X, X = x1...xN , be an input sequence, and Y , Y = y1....yN , be the label sequence for the input sequence. The conditional probability of Y given X is:

$$P(Y|X) = \frac{1}{Z(X)} \exp(\sum_{n=1}^{N} \sum_{k} \lambda_k f_k(y_{n-1}, y_n, X, n)) \quad (1)$$

where Z(X) is a normalization term, f_k is a feature function, which often takes a binary value, and λ_k is a learned weight associated with the feature f_k. The parameters can be learned by maximizing log-likelihood L which is given by

$$L= \sum_{i} \log P(Y_i|X_i) - \sum_{k} \frac{\lambda_k^2}{2\sigma_k^2} \quad (2)$$

Where σ2k is the smoothing (or regularization) parameter for feature fk.

- **Sentiment Analysis-** This analysis is based on the emotional quotient of the words arrangement in the sentence. To identify the sentiment of a string of text, from very negative to neutral to very positive. The ratings are done from 0 to 4, where 0 being the very good and 4 being very bad. The image below illustrated the four ways to Four ways to get from bad to good in three hops.

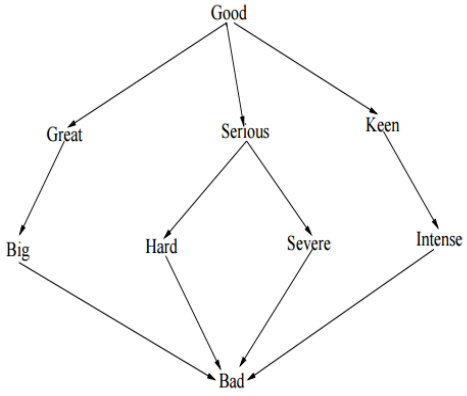

Figure5: *Flow for hoping from good to bad.*

The sentiment illustration uses the polarity and subjectivity logics for deducing the strength of text.

Subjectivity shows the ratio of sentiment to frequency of occurrence, while ***polarity*** indicates percentage of positive sentiment references among total sentiment references.

The below reference code which was customized for this project, exhibits the precision of execution.

```
from collections import namedtuple
from operator import attrgetter
from ..utils import ItemsCount
from .._compat import to_unicode
from ..nlp.stemmers import null_stemmer


SentenceInfo = namedtuple("SentenceInfo", ("sentence", "order", "rating",))


class AbstractSummarizer(object):
    def __init__(self, stemmer=null_stemmer):
        if not callable(stemmer):
            raise ValueError("Stemmer has to be a callable object")

        self._stemmer = stemmer

    def __call__(self, document, sentences_count):
        raise NotImplementedError("This method should be overriden in subclass")

    def stem_word(self, word):
        return self._stemmer(self.normalize_word(word))

    def normalize_word(self, word):
        return to_unicode(word).lower()
```

Figure6: *Snippet of python code for summarizer module*

```
import os, re, math, collections
import nltk.metrics
from nltk.classify import NaiveBayesClassifier

POLARITY_DATA_DIR = os.path.join('polarity-data', 'rt-polaritydata')
POSITIVE_REVIEWS = os.path.join(POLARITY_DATA_DIR, 'rt-polarity-pos.txt')
NEGATIVE_REVIEWS = os.path.join(POLARITY_DATA_DIR, 'rt-polarity-neg.txt')


def evaluate_features(feature_select):
    pos_features = []
    neg_features = []

    for line in open(POSITIVE_REVIEWS, 'r'):
        pos_words = re.findall(r"[\w']+|[.,!?;]", line.rstrip())
        pos_features.append([feature_select(pos_words), 'pos'])

    for line in open(NEGATIVE_REVIEWS, 'r'):
        neg_words = re.findall(r"[\w']+|[.,!?;]", line.rstrip())
        neg_features.append([feature_select(neg_words), 'neg'])

    print("len of positive features %d" % len(pos_features))
    pos_cutoff = int(math.floor(len(pos_features) * 3 / 4))
    neg_cutoff = int(math.floor(len(neg_features) * 3 / 4))

    print("pos_cutoff %d neg_cutoff %d" % (pos_cutoff, neg_cutoff))

    training_data = pos_features[:pos_cutoff] + neg_features[:neg_cutoff]
    test_data = pos_features[pos_cutoff:] + neg_features[neg_cutoff:]
```

Figure7: *Snippet of python code for Sentimental Analysis module*

The python code exhibited above uses the in-build library functions of python and with more of customization, the necessary output was formulated. The below screenshot shows the sample results got by running the above code.

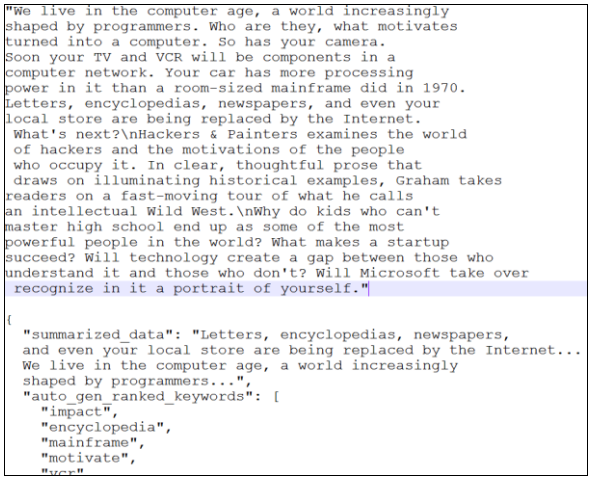

Figure8: *Snippet of output from Summarizer module*

The above results show that the huge text from Quora is summarized into considerably smaller text. The reduction rate is almost 75%. This makes the further processing easy and effective.

Then the AutoTag processing is done, where important keywords are extracted. These words are then tokenized, recognized using the names and analyzed sentimentally. The output of the NLP is processed using the ranking algorithm. Here each word is ranked based on the frequency of appearance, sentimental index [0-4], the reputation of the person who types in the comment. The reputation is manipulated using the number of upvotes his answers get, number people who follow him and number of questions he has answered in the relevant field. This relevant details about the ranking of the text are stored in the form of JSON format in the database, which later is summed up to a visualization logic.

Tokenization of text into a sequence of tokens is a process where the English component provides a PTBstyle tokenizer, extended to reasonably handle noisy and web text. The corresponding components for Chinese and Arabic provide word and clitic segmentation. The tokenizer saves the character offsets of each token in the input text. Thus this completes the entire decision logic of the text processing. The further discussion will focus on the visualization terminologies. The below Snippet shows the output from the NER and Sentimental unit.

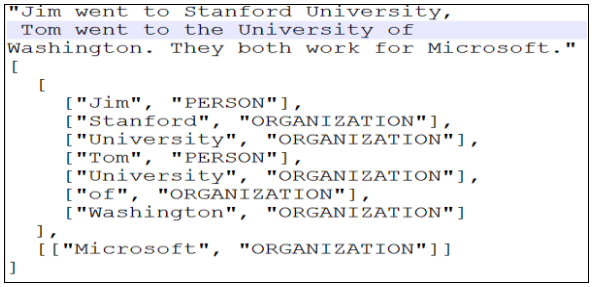

Figure9: *Snippet of output from NER module*

In the below output it is very evident that the algorithm scrutinized Person and organization. Here “Washington” was identified as Organization instead of Place because the preceding context was on the “University”, which is an Organization.

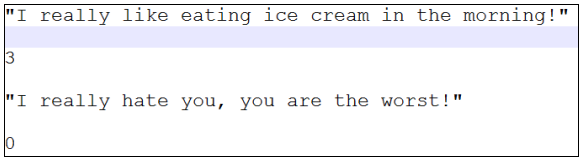

Figure10: *Snippet of output from Sentimental module*

Here the first sentence is in a positive note, where there were not negative notions. Second sentence has too many negative notions because of which it is rated as Very negative.

4 Data Visualization

As the suggested system is a web-based application, the visualization tools which has a good compatibility with these sort of applications were considered. The visualization for this paper was build using the technologies like D3.js, SVG, HTML and XML/JSON. The data processed by the NLP unit is stored in the JSON format. The JSON is very much lite and easy to parse and use it along with the tool. To add more, D3 is simple, robust and powerful visualization tool available for Web Applications. D3 uses HTML, SVG and CSS for customizing the required Visualizations. You can make the data interaction through the use of D3.js data-driven transformations and transitions. When a browser displays an HTML page, it shows an interactive graph from the tag hierarchy. This object graph is called the Document Object Model, or DOM

```
{
 "name": "flare",
 "children": [
  {
   "name": "analytics",
   "children": [
    {
     "name": "cluster",
     "children": [
      {"name": "AgglomerativeCluster", "size": 3938},
      {"name": "CommunityStructure", "size": 3812},
      {"name": "HierarchicalCluster", "size": 6714},
      {"name": "MergeEdge", "size": 743}
     ]
    },
    {
     "name": "graph",
     "children": [
      {"name": "BetweennessCentrality", "size": 3534},
      {"name": "LinkDistance", "size": 5731},
      {"name": "MaxFlowMinCut", "size": 7840},
      {"name": "ShortestPaths", "size": 5914},
      {"name": "SpanningTree", "size": 3416}
     ]
    },
```


Figure11: *Sample JSON file fed to the D3 js*

In the above file, each name is the JSON object and the associate children holds the value for which the visualization is intended to be displayed.

```
  var node = svg.selectAll(".node")
      .data(bubble.nodes(classes(root))
      .filter(function(d) { return !d.children; }))
    .enter().append("g")
      .attr("class", "node")
      .attr("transform", function(d) { return "translate(" + d.x + "," + d.y + ")"; });

  node.append("title")
      .text(function(d) { return d.className + ": " + format(d.value); });

  node.append("circle")
      .attr("r", function(d) { return d.r; })
      .style("fill", function(d) { return color(d.packageName); });

  node.append("text")
      .attr("dy", ".3em")
      .style("text-anchor", "middle")
      .text(function(d) { return d.className.substring(0, d.r / 3); });
});

// Returns a flattened hierarchy containing all leaf nodes under the root.
function classes(root) {
  var classes = [];

  function recurse(name, node) {
    if (node.children) node.children.forEach(function(child) { recurse(node.name, child); });
    else classes.push({packageName: name, className: node.name, value: node.size});
  }

  recurse(null, root);
  return {children: classes};
```

Figure12: *Sample HTML file coded in the D3 js*

Type of Visualizations:

- **Word cloud:** The words with more frequency are displayed bigger than the less frequent words. In other words, from this data, the more ranked word is displayed bigger in the word cloud.

- **Heat Map:** It's a graphical visualization for the variant scale from very positive to very negative. Precisely it is the visualization of data which are arranged in the form of matrix. The more the context of the answer is positive, the more it is brighter. Conversely, if the text is negative, the dimmer it is. The below view shows the sample output of heatmap got from a JSON input of a text.

Figure13: *An illusion for Word cloud*

```
var Heatmap = (function HeatmapClosure() {

  var Coordinator = (function CoordinatorClosure() {

    function Coordinator() {
      this.cStore = {};
    };

    Coordinator.prototype = {
      on: function(evtName, callback, scope) {
        var cStore = this.cStore;

        if (!cStore[evtName]) {
          cStore[evtName] = [];
        }
        cStore[evtName].push((function(data) {
          return callback.call(scope, data);
        }));
      },
      emit: function(evtName, data) {
        var cStore = this.cStore;
        if (cStore[evtName]) {
          var len = cStore[evtName].length;
          for (var i=0; i<len; i++) {
            var callback = cStore[evtName][i];
            callback(data);
          }
        }
      }
    };

    return Coordinator;
  })();
```

Figure14: *Sample code file for HeatMap*

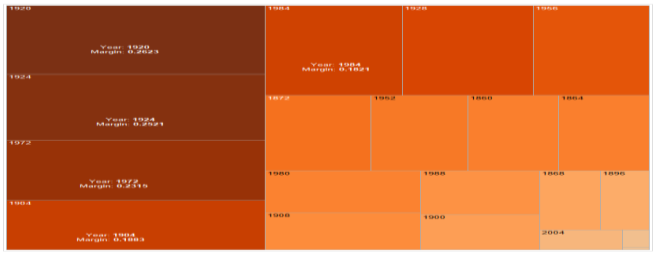

Figure15: *A view for HeatMap*

Network Graph: This is the feature where the question suggestions are made. If a user has

already asked a question and if there are similar question which has the more answers and the answers are highly ranked, then network map suggests those questions which relevant to the user. The user there by can click on the network and drilldown to the well framed questions. So, as the number of connections increases, the suggestion rate increases. The question suggestion looks the below way.

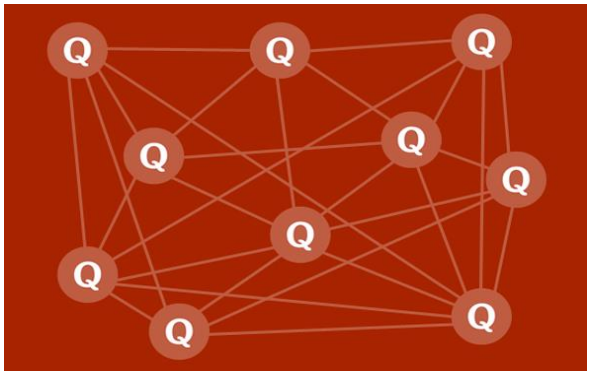

Figure16: *A view for Network Map*

5 Conclusion

In this paper, we have presented the design and usage of the Artificial intelligent system, an annotation-based NLP processing model for visualization of text for the better understanding of the context of the text. We have in particular tried to emphasize the properties that we feel have made it successful. The goal has been to make it as easy as possible for users to get started using the framework, and to keep the framework small, so it is easily comprehensible, and can easily be used as a component of the much larger system that a user may be developing. The broad usage of this system, and of other systems such as NLTK which emphasize accessibility to beginning users, suggests the merits of this approach. Moreover, the system can be extended for the customary five-star rating system, where any user can just click on the stars for giving their opinion on any product or any movie or anything for the subject of argument. This system will automatically read and sense the context of the comments and auto-populates the ratings based on the user's comments.

References

Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

Wenhui Liao and Sriharsha Veeramachaneni A Simple Semi-supervised Algorithm For Named Entity Recognition

Angel X. Chang and Christopher D. Manning. 2012. SUTIME: A library for recognizing and normalizing time expressions. In LREC 2012.

James Clarke, Vivek Srikumar, Mark Sammons, and Dan Roth. 2012. An NLP Curator (or: How I learned to stop worrying and love NLP pipelines). In LREC 2012.

Christopher D. Manning, Mihai Surdeanu, Jenny Finkel, John Bauer. 2014 The Stanford CoreNLP Natural Language Processing Toolkit.

https://d3js.org/

https://www.dashingd3js.com/why-build-with-d3js

https://pypi.python.org/pypi/beautifulsoup4

http://www.nltk.org/py-modindex.html