Moviements

Semantic Web Group Project

TDT4215

Web-intelligence

Group 12

Storvold, John-Olav

Sellæg, Jørgen Lien

de Miranda de Matos Lourenco, Ines

Abstract

The aim of this report is to take a better look at sentiment analysis. The proposed project consists on the creation of an application that works as a movie reviews database. It is possible to search for a specific review, create a new one, and know the scores associated with them. Instead of having to read all the reviews about a certain movie, users can with this app automatically know the average score of all the reviews of that specific movie.

Contents

1 Introduction

The internet became a very important platform where people can participate, express their opinions and views, share emotions and comments about a lot of everyday life aspects. It became a huge platform of opinions and sentiments' change. And due to this the concept of Sentiment analysis was introduced. The applications are countless, from

obtaining feedback from a certain product to important information regarding public opinion.

This analysis ends up being a win-win situation for both consumers and producers about products and services. If we know that a lot of people liked a certain product, then it must have some features worth paying for. We can also know which are its strengths and weaknesses, and the best and worst characteristics. For the producers it is beneficial in the sense that they can find out what should be improved in the future.

When it comes to the reviews, the idea of scoring them came up so that instead of reading all thousands of reviews of a product any user can just concern about the scores.

Given the possibility of choosing any theme inside in the semantic web topic, these are the reasons why we decided to mainly focus our attention on sentiment analysis.

We found a way to apply them to movies reviews because it seemed to be the most difficult of several domains for sentiment classification, reporting an accuracy of 65.83% on a 120- document set.

2 Theory

2.1 Sentiment analysis

It is important to distinguish between what is a rational evaluation (facts) and what is an emotional one (subjective). The base to do sentiment analysis is the interpretation of feelings. These can not only emotions but also attitudes and opinions.
We can want to classify users, texts, sentences/paragraphs, words, etc.
It is increasingly more important to have techniques that allow us to capture complex contextual phenomena. There are a lot of techniques that have been implemented in this field. The focus is now moving towards unsupervised approaches because of the huge amount of opinionated data.
A substantial number of sentiment analysis approaches rely greatly on an underlying sentiment (or opinion) lexicon. A *sentiment lexicon* is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative. They can be Semantic Orientation (Polarity-based) Lexicons or Sentiment Intensity (Valence-based) Lexicons.

2.1.1 Term-counting method

Individual words have what is referred to as prior polarity, that is, a semantic orientation that is independent of context; and that said semantic orientation can be expressed as a numerical value.

From here came the term-counting method that consists on having a dictionary with a list of words, each one with its own score. Applying the term-counting is a way to classify a review through its score, by summing the score of all the words present in the sentences of that review.

What it means is that the semantic orientation of terms and phrases can be used to determine the sentiment of complete sentences and reviews.

The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words or phrases in the document, using dictionaries of words annotated with the word's semantic orientation, or polarity.
This lexicon-based method performs well, and is robust across domains and texts.

2.1.2 Infinitive form

The method of counting positive and negative terms has a big boost if we start by transforming the terms from their current form into their base forms (lemmas). This increases the efficiency of the analysis due to the fact that without it we need to have all the word's forms in our dictionary.

This is for example the case of words such as *good*, *better* and *best.* Instead of having the three forms in our dictionary, we can simply tell the program that *better* and *best* are only different forms of the word *good*. What happens then is that when any of those words appear on the text to be analyzed, they get the same score as the infinite form of the verb (the only form that needs to exist on the dictionary).

2.1.3 Valence shifters

Valence, or in this case sentiment shifters, are very important to take in consideration because they change the polarity of a term. In this case, the polarity of a term represents whether it is positive or negative.

Not, no, none, never, nobody, nowhere, neither are some examples of sentiment shifters. If any of them appear before the word, then it should change the polarity of that word's score.

"This is *not* a good movie". If to the adjective *good* it is given a score of +3, then because of the presence of a sentiment shifter before the score would change to -3.

Sentiment shifter	+	positive word / negative word	=	negative word / positive word

2.1.4 Intensifiers

More than a polarity, words, phrases, sentences, or documents can have an intensity. It corresponds to the degree to which the word, phrase, sentence, or document in question is positive or negative.

Intensifier + positive word = more positive word	Word's score + % of intensifier
Intensifier + negative word = more negative word	Word's score - % of intensifier
Diminisher + positive word = less positive word	Word's score - % of intensifier
Diminuisher + negative word = less negative word	Word's score + % of intensifier

It can be said that if the word "Very" is an intensifier with 50% strength, then "Extremely" can be an intensifier with 100%.

For example: "I find this tool good and useful"
Let's say that "Good" has a score of +3 and "Useful" of +2.
Then the score of the sentence is 3 + 2 = +5.

Another example: "I find this tool good but **less** useful than yours"

If less is a 50% intensifier, the score of the sentence is now 3 + (50%*2) = 4, which makes sense being less than the previous one.

2.1.5 Difficulties

The analysis methods are pretty limited when it comes to some kind of situations.

First of all, it doesn't care about the context.

"This movie is *excellent* if you want to waste your money."

Another problem is that a word, such as an adjective, with a positive or negative meaning, doesn't always have and associated sentiment. It corresponds to a sentiment ambiguity.

"Is this a *good* movie?"

The opposite can also happen. A sentence without any sentiment word can also express sentiments. In this case it express a clearly negative sentiment.

"This movie shouldn't have been seen that much"

Irony and sarcasm are not recognized by the algorithms.

"I'm really *happy* that none of the movies encourage me to stay awake"

Finally, it is important to notice that a word can change sentiment depending on the context of the situation. For example, "catch" can have a negative meaning in the sentence "The movie had a catch", but neutral meaning in "I will catch the next train".

3 Tools

The following tools are what we use for our entire application implementation. It consists mainly of javascript libraries and tools with one exception that is running a python library to do the sentiment analysis.

3.1 *Node.js and Express.js*

Node.js is a javascript platform for the desktop. It allows developers to use javascript outside of a web browser. This means that programmers can develop servers, desktop-applications and other services on the computer.
With the introduction of html5 in 2014, usage of the http-protocol have significantly increased. A considerable percentage of this traffic is because of REST. REST is a pattern that allows developers to place data-sources easily reachable from any programming language with networking. Express runs on Node.js and is a REST-server is used to serve Mongoose database sources for the React client.

3.2 *MongoDB*

MongoDB is a cross-platform document-oriented NoSQL database. MongoDB offers simplicity and JSON-like documents with dynamic schemas which makes it a very solid database when used with JavaScript. We use MongoDB to store our movie reviews from either review submission or from getting the movie reviews from other sources online.
The schema for our review is fairly simple and is the following:
- Review(movie: String, title: String, review: String, score: Number, date: Date)
MongoDB automatically takes care of giving Review a proper ID, and all we have to do is to satisfy the constraints for requirements some of the attributes have. Movie, title and review is required, score is left untouched so that the system can process the review and add the score later and then date is filled with the todays date if left empty or saves the date given.
Moongose is our javascript bindings for MongoDB. It runs on Node.js and gives easy access to database schemas and it takes care of queries with a high-level API.

3.3 *React*

To develop frontend quickly with a lot of functionality developers use frameworks. Frameworks take care of generating html-pages/clients to present for the user with dynamic content. React covers the client functionality needed for this application. React is used to present the data provided from the REST-api to the user through the web application.

3.4 *VaderSentiment*

VaderSentiment is a lexicon and a rule-based sentiment analysis tool for python. It is a result of a research done back in 2014, and targets sentiment expressed in social media. This was a heavily influenced reason for why we switched to run our sentiment analysis using this tool instead of using "sentiment" previously which is performs an AFINN-based sentiment analysis. Before we made our switch to use vaderSentiment, we had problems properly reading the context of which the user submitted reviews where analyzed.

VaderSentiment is sensitive to the following language features:
- Polarity: Detecting the positive or negativity within the context
- Intensity: Use sentiment valence to highlight the polarity within the context.

VaderSentiment has been trained with a high number of submitted reviews and textual data from several sources. These sources include among other Twitter, Rotten Tomatoes, Amazon and NY Times.

3.5 *nlp_compromise*

nlp_compromise is a natural language processing tool and provides us with neat features we use to preprocess our textual data before we send it over for sentiment analyzing.
These features includes:
- sentence segmentation:
- expanding contracted words (I'll -> I will)
- normalizing a sentence: Turn the words into the infinitive.
- simplifying advanced words into simpler ones by using a defined lexicon.

4 Product description

4.1 *Goals*

Our App has 2 main pages: One where users are able to submit reviews to some movies they saw, and another one where users can search for reviews to a specific movie, or a lot of movies, and import them to the database.

Once imported, the app allows us to see the score of each review, and also to see the average score of all the reviews of a certain movie, which is useful for who wants a quick idea of the interest of the movie.

4.2 *Methods used*

We decided to use an unsupervised lexicon-based approach.

The first thing we decided to do was use the NLP (natural language processing), to simplify each sentence of the reviews.

Here we put every word on its root form, so for example for the case of verbs it puts all the forms like “spoke, have spoken, is speaking” in the infinitive, “speak”, which decreases drastically the amount of words needed in the dictionary. It also expands the contracted words, like “you’ll” to the form “you will”, and eliminates undesirable characters.

To the analysis itself we used VaderSentiment.

Firstly, we first have a list inspired by known sentiment word-banks like LIWC and General Inquirer, and smileys, slang, and words like “LOL” are added, because a lot of times movie reviews are not written in the most correct language. In fact there is contextual sparseness resulting from shortness of the text and a tendency to use abbreviated language conventions to express sentiments.

Then, “wisdom of the crowd” is used to give each words an intensity between "[–4] Extremely Negative" to "[4] Extremely Positive”, which consists in a lot of raters giving each word what they consider to be the most appropriate score.

After that every lexical feature that has a non-zero mean rating and standard deviation less than 2.5 is kept in the dictionary, while the others are eliminated.

In the end an amount of around 7,500 lexical features with validated valence scores are found, and adjusted with polarity and intensity.

Here’s an example:

Word	Score
Okay	0.9
Good	1.9
Great	3.1
Horrible	-2.5
☹	-2.2

Besides, the punctuation is also taken in consideration, which makes the feature “good!” has an higher score than “good”, and the same happens with capitalized words, which makes that “GOOD” becomes also better than “good”.

6 Results and discussion

The results we got were very close to the expected ones.
At first we tried only to implement an unsupervised approach using only a version of the AFFIN dictionary with scores to around 2400 words between -5 and +5. As expected, a lot of words were not included. If we wrote the word “fine” in a review then the score would be positive as expected, but if the word was instead “finest” then it wasn’t recognized, which obviously produced a lot of classification errors.

When we after introduced the treatment of the words to the infinitive form, a lot of classifications errors were immediately eliminated, since a lot of words started to be identified.

However, the accuracy had the most significant increase when the sentiment shifters and polarity shifters were implemented.

7 Future work

The biggest problems we noticed concerned the scores of the reviews. In fact, in the end of the algorithm we got scores between -1 and +1, but we wanted to convert them to a scale from 0 to 10, which is easier to be analyzed by the users.

In order to do this a logarithmic algorithm could be implemented.

Other improvements in the accuracy could be made by analyzing manually some system defects, and finally we could add a recommender systems algorithm, with user identification.

8 References

Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.