Institut für Informationssysteme
Abteilung für Verteilte Systeme
VU Advanced Internet Computing (184.269)

Project Topic 1 - Sentiment Analysis

Introduction

Your task in this project is to develop a simple cloud-based service for Twitter-based sentiment analysis[1]. Sentiment analysis is, broadly, the process of finding out (typically automatically) what the general feeling ("sentiment") of one or more Web communities (for instance, the blogosphere or the Twitter community) about a company or product is. This sort of analysis has become an increasingly relevant marketing tool in recent years.

Your service should provide two basic functionalities:

1. Prospective customers can *register* for your service, that is, they provide their company name for sentiment monitoring. You do not need to bother with other boring details, like payment information, at this point.
2. Afterwards, registered customers can *query* the aggregated sentiment for a specified time period. For simplicity, the output of your service should be a simple numerical value between 0 (people with pitchforks have been sighted striving towards the company headquarters) and 1 (people buy whatever the company CEO tells them to buy).

In this topic, the goal is to discover, implement, evaluate and tune actual sentiment classification algorithms. An overview over the field can be found in [2]. In this project, you should focus on analysing sentiments based on Twitter data, which poses some interesting challenges [1]. Test data should be retrieved live via the real Twitter API.

Outcomes:

The expected outcomes of this project are three-fold: (1) the actual project solution, (2) a brief paper that analyses the scientific state of the art in sentiment analysis in general and specific to microblogging services, and (3) two presentations of these results (paper and tool).

Project Solution

The project should be hosted on a public git repository, whereas the URL to the repository has to be submitted. We recommend either Github[2] or Bitbucket[3]. Every member of your team should have a separate Github or Bitbucket account. It is required to provide an easy-to-follow README that details how to deploy, start and test the solution. Test whether it is actually possible to build and deploy your solution according to your instructions on a random Ubuntu Linux, Windows or Mac OS X machine.

[1] `http://www.computerworld.com/s/article/9209140/Sentiment_analysis_comes_of_age`
[2] `https://github.com`
[3] `https://bitbucket.org`

Paper

The paper should analyse the scientific state of the art in sentiment analysis. Note that the paper is not the documentation of your tool – it should mainly discuss scientific papers related to this topics in the style of a seminar paper. However, put more focus on these approaches that are actually relevant / related to your solution. Good starting points for finding related scientific papers are the works cited in this text, Google Scholar[4], IEEEXplorer[5] or the ACM Digital Library[6]. Use the ACM 'tight' conference style[7] (two columns), and keep it brief (3 pages). You do not necessarily need to install a LaTeX environment for this - you can use writeLaTeX[8], a collaborative paper writing tool as well.

Presentations

There are two presentations. The first presentation is during the mid-term meetings (Nov. 28th and Nov. 29th), and should cover at least the tasks of Stage 1 (see below), the second presentation is during the final meetings (Jan. 30th and Jan. 31st) and contains all your results. Every member of your team should participate in either the first or the second presentation. Each presentation needs to consist of a regular e.g., Powerpoint part and a demo of your tool. Carefully think about how you are actually going to demonstrate your tool, as this will be part of the grading. You have 20 minutes per presentation (strict).

Grading

A maximum of 50 points are awarded in total for the project. Of this, up to 25 points are awarded for the tool, up to 10 points are awarded for the paper, and up to 15 points are awarded for the presentations. All gradings will necessarily be subjective (we judge the quality and creativity of solutions, presentations, and papers).

Deadline

The hard deadline for the project is **January 29th, 2014**. Please submit a link to your solution repository, deployment instructions, the paper, and the presentations as a single ZIP file via TUWEL. The submission system will close at 18:00 sharp. Late submissions will not be accepted.

Test Cloud Infrastructure

This year, we have access to a grant for Amazon Web Services[9] (AWS), which you can use to deploy and test your solution. Send an email to `aic13@dsg.tuwien.ac.at` if you wish to get access to the AWS cloud environment.

Stage 1

Broadly, sentiment analysis for Twitter data encompasses four steps (see Figure 1). Firstly, (relevant) data is loaded, either directly from Twitter or from a local cache. Then each tweet needs to be preprocessed (this includes stemming and stop word removal) and analyzed. In the simplest case, this can include a ternary classifier that categorizes each tweet as either, positive, negative, or neutral. Finally, after all individual tweets are categorized, a summary sentiment needs to be built by aggregating all individual classifications. Your task in Stage 1 is now to build a framework that brings this abstract process to life.

[4] `http://scholar.google.at/`
[5] `http://ieeexplore.ieee.org/Xplore/home.jsp`
[6] `http://dl.acm.org/`
[7] `http://www.acm.org/sigs/publications/proceedings-templates`
[8] `https://www.writelatex.com`
[9] `http://aws.amazon.com`

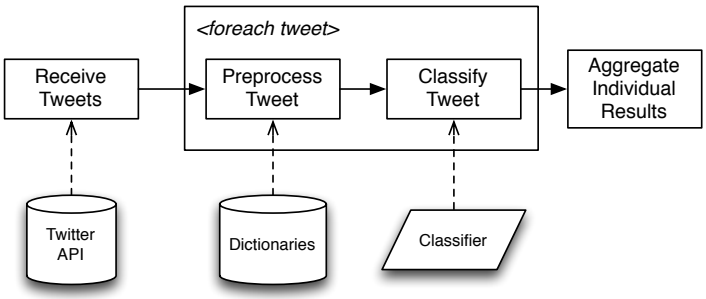

Figure 1: Abstract Twitter Sentiment Analysis Process

Tasks:

1. Interact with the Twitter API (e.g., via Twitter4J[10], if you are using Java) to download relevant tweets to analyse. How you select which tweets are relevant in the first place is up to you, you can e.g., restrict by location, language or use specific search terms. However, keep in mind that Twitter enforces a rate limit on its API, so it may make sense to cache tweets locally to prevent them from being re-downloaded unnecessarily.

2. Implement a preprocessing approach for your tweets. The traditional way to implement stemming and stop word removal is to use pre-defined dictionaries (which are, of course, language specific), which you can find on the Internet. However, keep in mind that Twitter works a little differently than longer and more formal texts. For instance, :), :/, etc. should probably not be removed from a tweet. Further, hash tags (words starting with #, e.g., #nrw13) often carry specific semantics.

3. Building the actual classifier is in theory possible with any supervised or unsupervised machine learning approach, but, clearly, some approaches can be suspected to work better than others. In this stage, focus on getting a reasonable result with a single machine learning approach, for instance Support Vector Machines (SVMs). A summary from the tourism domain can be found in [3]. You should not just integrate an existing sentiment analysis tool (e.g.,[11]) into your solution, but it is ok to use an existing tool to generate the "ground truth" for supervised learning. Furthermore, it is perfectly ok (even expected) that you do not implement the basic machine learning model yourself, but build up existing implementations, such as R[12] or WEKA[13].

4. Finally, aggregate the results. The main decision point here is whether each individual tweet should carry the same weight, or whether certain tweets (because of higher confidence, higher sentiment, closer geographic location, more recent date, ...) should be more important than others. This is again a point where reading existing scientific papers should provide some inspiration.

Stage 2

Stage 2 of this topic is mainly about refining what you did in Stage 1. In terms of new features, you should now add wrap your sentiment analysis process into a SOAP-based or REST Web service, and also add a minimal GUI for demonstration purposes. Furthermore, you should now focus on experimenting with different implementations of your classifier, and improve its performance.

[10] `http://twitter4j.org/en/`
[11] `http://text-processing.com`
[12] `http://www.r-project.org`
[13] `http://www.cs.waikato.ac.nz/ml/weka/`

Tasks:

1. Wrap your sentiment analysis application from Stage 1 into a simple Web service (your choice of REST or SOAP-based). Essentially, the operations of your service should be as introduced in the Introduction.

2. Additionally, implement a simple GUI (for instance a Web interface). The GUI does not need to be pretty, but should be reasonable for testing and demonstrating your service.

3. Finally, it is time to revisit the Classify Tweet step in Figure 1. Your goal in this stage should be to experiment with different algorithms and implementations, and see to what extend you can improve your original implementation. Also look at how having more or less training data (in case of supervised learning) influences the quality of your classifications. Your service has to allow for a parameter that allows users to specify an algorithm / configuration to use as parameter. Make sure to demonstrate and explain these experiments as part of your final presentation as well.

References

[1] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! In Lada A. Adamic, Ricardo A. Baeza-Yates, and Scott Counts, editors, *ICWSM*. The AAAI Press.

[2] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.

[3] Qiang Ye, Ziqiong Zhang, and Rob Law. Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert Syst. Appl.*, 36(3):6527–6535, April 2009.