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

**Project Topic 2 - Crowdsourcing**

# 1 Introduction

Your task in this project is to develop a simple service for sentiment analysis[1]. Sentiment analysis is, broadly, the process of finding out 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 will use the plaintext articles that are periodically issued by the well-respected Yahoo! Finance portal[2]. These reports often mention companies and products positively or negatively, and can serve as indicators of general public opinion. As it is quite difficult to accurately extract sentiment information of such plaintext articles, we will use the crowdsourcing [2] paradigm to let human "workers" identify mentions and classify them as positive or negative.

You should implement two simple functionalities for your service:

1. The main task of your service is to *parse* the articles emitted by Yahoo! Finance and to generate a knowledge base of company and product mentions. This should be achieved via Crowdsourcing.

2. Afterwards, customers can *query* this knowledge base via a service interface (e.g., REST or SOAP). For simplicity, we will ignore aspects such as customer registration, security, billing, scalability, etc. in this project and focus on the crowdsourcing aspects instead.

Crowdsourcing is based on the assumption that there are tasks that are almost impossible to solve algorithmically, but which can still be easily solved by any human (even without specific job training). Examples include complex image or text classification problems. In crowdsourcing, applications can post small "tasks" to an anonymous "crowd" of human workers (including smallish monetary rewards for solving the tasks). Workers will then select tasks they like (or which are well-paid) and solve them. In that sense, crowdsourcing allows us to seamlessly integrate human intelligence into our applications.

In this project, you should use the MobileWorks[3] crowdsourcing platform. Other than that, you are free to implement your service using any programming language / platform you feel is suitable for the task.

# 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 for crowdsourcing applications, and (3) two presentations of these results (paper and tool).

[1] `http://www.computerworld.com/s/article/9209140/Sentiment_analysis_comes_of_age`
[2] `http://finance.yahoo.com`
[3] `https://www.mobileworks.com`

## 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[4] or Bitbucket[5]. 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.

## Paper

The paper should provide an overview about the scientific state of the art in crowdsourcing. Note that the paper is not the documentation of your tool – it should discuss scientific papers related to this topic in the style of a seminar paper. Good starting points for finding related scientific papers are the sources cited in this text, Google Scholar[6], IEEEXplorer[7] or the ACM Digital Library[8]. Use the ACM 'tight' conference style[9] (two columns), and keep it brief (3 pages). You do not necessarily need to install a LaTeX environment for this - you can use writeLaTeX[10], 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 has to participate in either the first or the second presentation. Each presentation needs to consist of a regular e.g., Slides part and a demo of your tool. Carefully think about how you are actually going to demonstrate your solution, as this will be part of your 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[11] (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.

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

## Stage 1

In the first stage, you should implement the two basic functionalities mentioned above. For the first functionality, you will need to periodically download articles from Yahoo! Finance, split them into solvable human intelligence tasks, post them to the crowd, collect results and store them for later retrieval (e.g., in a database). MobileWorks provides you with means to cluster tasks into projects and to receive callbacks upon finished tasks. Make sure to make good use of these possibilities. Furthermore, make sure to use the sandbox mode of MobileWorks (you don't want to post tasks to the actual crowd!). Reading over existing scientific use cases of crowdsourcing (e.g., [4, 1]) may serve as inspiration and/or a starting point for devising your own solution.

### Tasks:

1. Design your crowdsourcing architecture. What types of tasks do you use? Is there a defined order between tasks, or can they be solved independently? What input does a worker need for each task? Typically, you want to give the same task to multiple workers and aggregate results, however, then you have to think about conflict resolution. Furthermore, think about what to do about tasks that are not taken up by any worker.

2. Implement your crowd-based sentiment mining solution based on a platform of your choice and MobileWorks. Also define one or more simple sentiment metric(s) based on the generated mentions database.

3. Write a simple service that allows customers to query this sentiment data (for instance via SOAP or REST).

## Stage 2

In the second stage, you should extend the service with a simple Web-based user interface. Furthermore, extend your crowdsourcing solution by also taking into account quality management of human workers. Keep track of the performance of your workers, and block the ones that are often underperforming. Furthermore, think about dynamically pricing your tasks. Dynamic pricing should have the goal of keeping tasks cheap that will be solved quickly anyway, and increasing the price of tasks that are prone to being ignored by workers. Some experimental findings that may be of relevance to your problem (or which are at least of interest to the general topic of pricing crowdsourcing tasks) can be found in [3]. Finally, some tasks may never be taken up by any worker. Make sure to revoke those tasks again once they become irrelevant or outdated.

### Tasks:

1. Start with the updates to your crowdsourcing model. Firstly, think about how you can measure the quality of your human workers. Once you have a metric for this, blocking notoriously bad workers is trivial over the MobileWorks API.

2. Afterwards design your dynamic pricing solution. Think about what your general approach is going to be, and implement it.

3. Add a "garbage collection" mechanism that tracks which tasks are never being picked up, and removes them.

4. Finally, build a simple Web interface for your whole service. The Web interface should be both, a management interface for inspecting the sentiment data and the quality of your workers, as well as a GUI for issuing queries to your service and visualize the results (e.g., via diagrams or plots). You may use your choice of platform for building this Web interface.

# References


[1] Omar Alonso, Daniel E. Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15, November 2008.

[2] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, April 2011.

[3] John Joseph Horton and Lydia B. Chilton. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, pages 209–218, New York, NY, USA, 2010. ACM.

[4] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM.