Sentiments in Sustainability Data Collection: Understanding User Sentiment in Collaborative, Social Tagging Environmental Sciences Platforms

Nikita Sanjay Panchariya

Follow this and additional works at: https://digitalcommons.montclair.edu/etd

Part of the Computer Sciences Commons
Sentiments in Sustainability Data Collection: Understanding User Sentiment in Collaborative, Social Tagging Environmental Sciences Platforms

by

Nikita Sanjay Panchariya

A Master’s Thesis Submitted to the Faculty of Montclair State University
In Partial Fulfillment of the Requirements
For the Degree of

Master of Science

May 2016

College of Science and Mathematics

Department of Computer Science

Thesis Committee:

Dr. Katherine Herbert
Thesis Sponsor

Dr. Rashmi Jain
Committee Member

Dr. Jeffy Fan
Committee Member
ABSTRACT

Many scientists throughout the world are exploring how the data can be analyzed and used for sustainability purposes. With these investigations there is significant need to collect and explore the data with mobile collaborative tools to enable real time interactions among the interested users and the researchers. The principle objective of this study is to explore an application of sentiment analysis on the collected heterogeneous data in the form of text, images, and location details that were collected using the Geotagger application and observe and infer users’ reactions and opinions of the geographical data they collect about.
SENTIMENTS IN SUSTAINABILITY DATA COLLECTION:
UNDERSTANDING USER SENTIMENT IN COLLABORATIVE, SOCIAL
TAGGING ENVIRONMENTAL SCIENCES PLATFORMS

A THESIS

Submitted in partial fulfillment of the requirements

Master of Science in Computer Science

By

NIKITA SANJAY PANCHARIYA
Montclair State University
Montclair, NJ
2016
Acknowledgments

I would like to thank my advisor, Dr. Katherine Herbert, for her valuable guidance, indispensable advice, eminent support and patience during my research. I would also like to thank my dissertation committee members Dr. Jerry Fails and Dr. Rashmi Jain for their support and benevolence. I would also like to acknowledge Dr. Emily Hill for her support in my work. I would also like to thank Andrew J. DeStefano for his prominent help throughout. I would like to recognize the support of the consultants at the center for writing excellence at Montclair State University; Meagan DeJong, Ann McKinstry Micou, Camilla Finch, Emily Lagg, Emily Stypolkowski, Norman DeFilippo, and Peter Florek in my writing. I express sincere appreciation to Dr. Constantine Coutras, the Chairperson of the Department of Computer Science for his kind guidance and support. Finally I would like to thank my family and friends for their relentless support and motivation towards my research.
Table of Contents

Abstract ........................................................................................................................... 1

Acknowledgments ........................................................................................................... 4

1. Introduction .............................................................................................................. 7

2. Thesis Description .................................................................................................... 8

3. Impact ....................................................................................................................... 9

4. Hypothesis .............................................................................................................. 10

5. Related Work .......................................................................................................... 11

5.1 Sustainability Science and Data Mining ........................................................ 11

5.2 Sentiment Analysis and Opinion Mining ....................................................... 12

5.3 Social Media in Data Mining and Decision Making ........................................ 13

5.4 Application and Techniques of Sentiment Analysis ....................................... 14

5.5 Exploiting Emoticons in Sentiment Analysis ................................................. 17

6. Methodology and Tools .......................................................................................... 18

6.1 Proposed Technique – Lexicon based sentiment analysis .............................. 18

6.2 Survey of Tools for analyzing the existing data ............................................. 19

6.3 Geotagger data analysis using sentiment analysis tool Semantria ................ 20

7. Geotagger formative participatory design session with some direct feedback with regards to different possible user interfaces .............................................................................. 22

7.1 Participatory design session setup ................................................................. 22

7.2 Participatory design session conditions ......................................................... 22

7.3 Participants ...................................................................................................... 22

7.4 Methodology ................................................................................................... 22

7.5 Measures .......................................................................................................... 23

7.6 Results ............................................................................................................. 23

7.7 Participatory design session questionnaire and suggestions / comments from the users ......................................................................................................................... 25
7.8 Participatory design session user's suggested prototypes and overall suggestions

8. Implementation

8.1 System Design

8.2 System Implementation

9. Future Scope

10. Conclusion

Bibliography

Appendices

Code

Tag Edit

Tag View

Add Tag XML
1. Introduction

Technology has become an integrated part of modern culture. With technology the field of human computer interaction is also an ever-increasing area. This proliferation of technology has impacted society as a whole, but in particular children. In the United States it has been estimated that 66% of 8-18 year olds have cell phones, and that children in this age group interact with entertainment media for approximately 7.5 hours each day [1]. This increase in the use of technology has impacted the children in means of exploring the world outside. The interaction of the children with the natural world around them has been reduced. Research is ongoing to overcome this kind of scenario and where something should be done to change this dramatic situation. There should be some technology that motivates the children to go out and explore the natural world and bring them together. Geotagger is a collaborative environmental inquiry platform that enables children and adults alike the opportunity to observe the world around them, document that observation, share it, and encourage discussion around that tagged item of interest. Users can have conversations about tags through a comment feed about each tag. Tags can be grouped into adventures so they can be configured for a specific purpose, like investigating certain plant types or habitats, or creating a treasure hunt [1].

The field of data analysis continues to grow as an ever-increasing volume of data is being collected and made available on the web. This vast expanse of data is termed as ‘Big Data’. Big data is comprised of both structured and unstructured data, which is impossible to analyze using the traditional software techniques. Extracting some structured information from the big data and analyzing the information with regards to sustainability is a major area of research. Sentiment analysis is one of the methods used for identifying and extracting subjective information from unstructured heterogeneous data sets [6]. The principle objective of this research is to explore an application of sentiment analysis on the collected heterogeneous data in the form of text, images, and location details collected using the Geotagger application and observe and infer users’ reactions and opinions of the geographical data they collect.
2. Thesis Description

The research mostly encompasses multiple disciplines of Data Analysis and Sciences, such as Big Data, Sustainability Science and Sentiment Analysis. The purpose of this research is to survey the available data sets and make a decision about the users' feedback. Further, analyze the data sets to check the sustainability concerns for a specific geographical region, then integrate functionality to the existing application Geotagger to collect the data. Geotagger is an application that was originally aimed at children but has evolved into a platform for collaborative data sharing and discussion. It allows users to take pictures and create geo-tags with description information to share with other children and create a social learning experience. For example, a person could take a picture of a pine tree with a geo-tag of where they found it along with a short description, then other people that log in to the application will be able to view other people's tags. They can also make comments on that tag, and can add tags of their own, which in turn can trigger a collaborative dialog via comments on that tag.

The objective of the whole project is using technology to encourage people to observe the natural world around them and to discuss the information with peers and colleagues. As users interact with Geotagger, they are actively engaged in collaborative, constructive, and generative activities. The project also has a children's research team involved, who give feedback on the system modules. Geotagger allows the users to create small tags by taking pictures of the things they observe in their surroundings and add a short description of what they think about the tags, then the other users can log-in to the application and view tags and comment on it. Now there is also a module for sentiment analysis where the user via an emoticon will be able to express his/her feeling about a certain tag.

While the overall goal of this research is broad, after the implementation of the functionality into geo-tagger the collected data will be analyzed through the sentiment analysis techniques to analyze the feedback from the user according to their opinion against the location details to check the sustainability concerns for a specific region.
3. Impact

The technology is a very important aspect and has a large impact in the present day society not only with adults but also children. The technology should be able to bring children together to share, analyze and create with their friends or their peers. The Geotagger mobile application is an approach to bridge this gap especially to help children make a choice in between nature and technology. It differentiates itself from various citizen science approaches with limited data collection and analysis: providing social interactions about the observations they made, along with being able to rate the tags and decide if it is interesting for them. Using Geotagger the participants can tag their interested observations about their surroundings and real world things and view what others are interested in, followed by a conversation about their interests and also rate the pre-existing tags. This can be very useful for a specific purpose, like collecting data about any specific plants or anything else in the habitat, treasure hunts, etc. as the tags can be configured.

The principle objective of this study is to explore an area of sentiment analysis on the collected heterogeneous data in the form of text, images, and location details collected using the Geotagger application and observe and infer users’ reactions and opinions of the geographical data they collect. The sentiment analysis of the previously collected data gives an idea about the necessity to implement the mobile solution for rating purpose. The module with emoticons in the mobile app and the rating module on the website make it quicker for the user to express his/her opinion about a certain tag. The mobile solution for gathering the rating gives the user a better perspective to express his/her feelings using the emoticons. As the Geotagger application is mainly intended for children in helping them to improve their technical, social and cognitive skills, adding emoticons gives them a visually clear idea about being able to express their opinion in a fun way. One key aspect of this tool is that since its design is highly collaborative with special attention paid to the HCI aspects, it enables users, especially children, to participate in the field. Therefore, it presents a flexible mobile application that can allow interactive data collection and share activities to occur in real time, with little restriction on the age appropriateness or education level of the user.
4. Hypothesis

In today’s era of technology, the field of data analysis is growing as a large volume of data is being collected through the web. The large volume, velocity and veracity of data is termed as ‘Big Data’, which is difficult to manage using traditional database systems. According to the United States Census Bureau there are 19.9 million people living in the New York metro area [13] and many scientists throughout the world are exploring how the data can be analyzed and used for sustainability purposes, such as in a situation like Hurricane Sandy, where there are various sustainability issues like Hurricane evacuation and sheltering, human and infrastructure systems for hurricane evacuation, etc. [14]. Therefore, when we talk about crowdsourcing in sustainability, one such user friendly and crowdsourced application could provide a scenario where these 19.9 million concurrent users can provide an opportunity for further research for handling these situations in a better way. Big data comprises of both structured and unstructured data, which is impossible to analyze using the traditional software techniques. Extracting some structured information from the big data and analyzing the information with regards to sustainability is a major area of research today [11].

Sentiment analysis is one of the methods used for identifying and extracting subjective information from the unstructured heterogeneous data sets. This study explores the application of sentiment analysis on the collected heterogeneous data in the form of text, images, and location details. The purpose is to observe the reactions about the geographical data they collect about a specific plant species, the opinion of end users as they interact with the other users or just the users’ opinion about a specific species of plant nearby to them [6].

The focus of this research is to understand sentiment analysis application in Big Data by analyzing the data. Conclusions are drawn from that analysis with regards to the impact of the data on the users, whether they liked to be a part of the project to express their opinions using the tool we implemented, and understanding the concerns in a specific region with the information collected from the users and their location details. The sequence of my work goes as follows:
i. We conducted a literature review to survey the field of sentiment analysis in Big Data and Crowdsourcing. To begin with, we conducted a survey regarding how sentiment analysis can be applied to the unstructured data sets and also a survey regarding a crowdsourcing technique, which we can use for my data analysis.

ii. We analyzed the existing data using a text analytics tool in order to map the necessity of the new sentiment analysis research and module in the existing project.

iii. We integrated additional functionality for analysis of the data to an existing system. In order to do this, we added new functionality to the existing sustainability project, Geotagger, to support the gathering of sentiment information with regards to tags.

5. Related Work

5.1 Sustainability Science and Data Mining

This article "Social Observation for Sustainability Science about Water" by J.B. Braden discusses the challenges the scientific community has to face while gathering the data for the different aspects of sustainability issues, such as magnitudes of time, space and function, multiple interests, failures etc. During the research, the author realizes that the data collection of existing program for sustainability is not enough and new programs are needed. Sustainability programs are mostly localized due to the complexity of the interaction between human and nature, while there are many other issues which require research at larger scale; examining the issues regarding climate and how attitudes influences the evolution of regulatory institutions. To design the information program about water, attitudes and behaviour can be important factors in linking the environmental conditions to perceptions. There were different types of data sources from where the water samples were collected nationwide; Wadeable Stream Assessment (WSA) collected 100 water quality indicators at 2,042 locations nationwide, General Social Survey (GSS) collected the attitudinal, behavioural and social involvement of data and the National Survey on Recreation and the Environment (NSRE), which gives data relevant to household [18]. After geospatially
matching the social and water information, the observations were as follows: the WSA and GSS observations were from 129 same counties, while only 4 from the same census tracts. After analysing two attitude questions and one behavioural question, a significant overall effect and few regionally specific effects were noticed. This evidenced the regional difference in water attitude. The WSA and NSRE data was used to study the water quality based on the zip-code scale, which illustrated various limitations of the data sets [18]. One of the limitations was that a lot of information available cannot be used because of protection and privacy issues. The core of sustainability science is that the social and the natural systems must be related; current data collection programs do not support insights about important information.

5.2 Sentiment Analysis and Opinion Mining

Sentiment analysis is a field of research where the computational treatment of human opinions is dealt with many sophisticated algorithms. Investigation of SA (sentiment analysis) applications is interlinked to the areas such as transfer learning, emotion detection, and building resources. This article “Sentiment analysis algorithms and applications: A survey,” by W. Medhat explains the survey of recently proposed algorithms', enhancements and their applications. Sentiment Analysis (SA) or Opinion Mining (OM) is a computational analysis of human attitude and emotions. Some researchers identify SA and OM as interchangeable with slightly different meanings. Sentiment analysis is useful in identifying and analyzing the sentiments and emotions of a text, whereas opinion mining interprets the opinion of a user related to an entity [10].

Sentiment analysis uses three-step classification process to identify the complete emotions of a document. In these three levels, the analysis would identify the expression of a product review in document level, sentence level, and aspect level. A document in three levels would be identified as either subjective or objective as well as either positive or negative opinion. Document level and sentence level SA would not give that much of difference in the analysis. Therefore, aspect level SA is needed to obtain the details, which are needed for many applications. Product reviews are a very good source for obtaining data and are used for sentiment analysis. Business decisions can be made using the analysis of these product reviews. Sentiment Classification techniques can be roughly divided into three categories that are the
machine learning approach, the lexicon-based approach and the hybrid approach. The machine learning approach relies on the famous ML algorithms to solve the SA as a regular text classification problem that makes use of syntactic and/or linguistic features, which are used to solve a regular text classification problem with the help of famous ML algorithms [10].

The lexicon-based approach analyses the text and finds an opinion of a user. Positive and negative opinion words are employed to express desired and undesired states, respectively. An opinion lexicon, which is the combination of opinion phrases and idioms, is also employed in sentiment analysis tasks. An opinion word list can be compiled in three different approaches. They are manual approach, dictionary based approach, and corpus based approach. As manual approach is time consuming, it is combined with the other two automated approaches to get the results efficiently. Dictionary based approach cannot find opinion words with domain and context specific orientations. Corpus based approach is useful in solving problems of finding opinion words with context specific orientation. Methods in this approach use a syntactic pattern or the pattern that finds other opinion words in a large corpus. There are other techniques which do not come under either ML approach or lexicon-based approach. One of those approaches is Formal Concept Analysis (FCA). This is a mathematical approach which is used for structuring, analyzing and visualizing data using the notion of duality. This article concludes that more research needs to be done for the enhancements of SC algorithms. More work needs to be done in sentiment analysis as there is a lack of resources in the languages other than English. Social network sites and micro blogging sites are playing a major role in expressing people’s opinions regarding a topic or product. Deeper analysis needs to be done to evaluate these data. Researchers are being attracted more to do research on SA process using NLP tools [10].

5.3 Social Media in Data Mining and Decision Making

The goal of this article “Hot or not: a qualitative study on ecological impact of social media & fashion consumption,” by Y. Pan, was to investigate the influence of social media on people in making their decisions about fashion consumption and how this affects sustainability. The author describes that by saying “Fashion” he means a lot more than “Fashion Industry” which relates to the apparel. Here Pan is referring to
luxury items, cosmetics, using bottled water and those items of living which aim
towards signaling social belonging rather than just satisfying the physical need [19].
The study was conducted using basic interview technique and analyzing the text of a
popular website which helps people in making their decisions while buying the items.
The findings state that almost all the participants in the study had intrinsic motivation
to buy the digital equipments like cell phones and laptops. The participants stated that
they consider buying a new one only when the old one is broken or hard to use. Few
participants mentioned that they bought the items for self identification or hobby
purposes. This article also explains the other factors which motivate the users for
buying the products; these extrinsic factors include social aspects, monetary aspects,
performance and material. Considering the ecological impact, the participants spoke
about e-waste and recycling. It was interesting that some participants from India and
China mentioned that they don’t consider replacing the item unless its broken or worn
out, and they can make the best use of whatever they have. In order to understand the
social media impact on the decision making, four different subsets of social media is
used in this article. First, Online Review: All the participants mentioned that online
review helps a lot in making the decision whether or not to make a purchase. The
second social media subset was emails; the participants used emails only when they
had someone known to advise them. The third one was Micro blogging, and the
participants highly recommended this. The fourth one was the social media tools like
Twitter and Facebook which also highly affected the decision making. The author
addresses a problem where social media promotes digital consumption and vice versa,
but it lacks the consideration of the ecological aspects in it. The author suggests that
there should be a model where sustainability aspects should be considered in the
social media while making the decisions about the digital consumption [19].

5.4 Application and Techniques of Sentiment Analysis

Sentiment analysis is “a task of finding opinions of authors about specific entities.”
In this article “Techniques and applications for sentiment analysis”, R. Feldman,
reports sentiment analysis as one of the newly evolving areas of research in computer
science, and outlines the decision making process which depends on the opinions of
others. By giving a simple example, the author explains how the reviews written by
others affect the decision of the buyer while shopping online. Social media is the
strongest resource for ‘sentiments’; all these snippets of text are claimed to be mines
for the companies to monitor the feedbacks for their products. To look into the topic deeply, the author classifies the sentences in two categories subjective and objective. Objective class contains the facts about the information, whereas subjective class contains the opinions, beliefs, and views about something. Focusing on the subjective sentiment analysis, the author specifies 5 different sentiment analyses: document level, sentence level, aspect-based, comparative and sentiment lexicon acquisition [20].

Document level sentiment analysis: In this technique, it is assumed that the document provides an opinion about a specific issue. It has two main classifications, supervised and unsupervised analysis. Supervised analysis involves analyzing a finite set of classes, for e.g. positive and negative opinions. Unsupervised analysis determines the semantic orientation of the specific phrases in the document. The two methods used for the selection of phrases based on predefined patterns and lexicon of sentiments of words and phrases.

Sentence level sentiment analysis: In order to have details of the opinions expressed in the multiple sentences of documents about the same entity, sentence level analysis is introduced. Two assumptions are made to analyze the sentence: the identity of the entity is known and the sentence has a single opinion, which makes it easier to split the sentence into phrases. Before analyzing, it is necessary to determine whether the sentence is subjective or objective and then the supervised and unsupervised techniques are used to analyze subjective phrases. The challenge at this level is to detect sarcasm in the sentences.

Aspect-based sentiment analysis: When people talk about a specific entity and express their opinions, they also have different opinions regarding the different aspects of the entity. Usually, this is seen in the reviews of the products where people talk about the different attributes related to the product. Opinions about some aspects might be positive while some might be negative, and these cannot be analyzed as a whole review. Recognizing the sentiments in a document and the aspect to which they relate is aspect based sentiment analysis. One approach which is most commonly used when solving this research problem is extracting all the noun phrases and keeping only those noun phrases whose frequency is above some pre-determined threshold, or reducing the noise in the noun phrases.
Comparative sentiment analysis: It is not necessary that the user will always talk about the same entity; sometimes the user gives a comparative feedback with another entity. Identifying the sentences which have comparative opinions and extracting the preferred entity from it, is the goal of comparative analysis. The author describes the different words for comparison, such as comparative adjectives and adverbs: ‘more’, ‘less’, words ending with ‘er’; superlative adjectives and adverbs like ‘most’, ‘least’, words ending with ‘est’; additional phrases such as ‘favor’, ‘exceed’, ‘prefer’, ‘than’, ‘superior’, ‘inferior’, ‘number one’, etc. Sentiment lexicon acquisition: The article expands three different approaches for the acquisition of lexicon: manual approach – hand coded lexicon, dictionary based approach – expansion of words, is done using the resources available; corpus based approach – the set of seed words, is expanded using large corpus of documents. The manual approach, being too laborious, is not practicable. In the dictionary based approach, the sets of words are expanded, using the synonyms and antonyms from resources like word net and then checked whether the sentiments are positive or not using algorithms. Identifying additional adjectives that have a consistent polarity, as a set of seed adjectives, is sentiment consistency, which can be used in corpus based approaches. A set of connectors like ‘and’, ‘either-or’, ‘neither-nor’, ‘or’ are used to find the connected adjectives and then to decide if one adjective is positive, if the other is positive or negative [20].

Lastly, there are many open research issues in the field of sentiment analysis.

i. Need for better modelling of compositional sentiment at the sentence level, more accuracy in calculating the sentiments based on words.
ii. One entity might have different names in the document; this issue is automatic entity resolution, which is yet to be solved.
iii. When several entities are discussed in a document, it is difficult to find something relevant.
iv. Identifying sarcasm is not yet integrated with sentiment analysis systems.
v. Noisy Text [20].
5.5 Exploiting Emoticons in Sentiment Analysis

Sentiment is analyzed using the emoticons used by the users. Now-a-days, the users can express their opinion by using blogs, reviews, posting messages on discussion forums or publishing their opinions on social networks where there is lots of information on the user's sentiments which can be analyzed [15].

If the user is communicating through the plain-text using a computer as a medium, the sentiments of visual cues are lost, whereas while communicating face-to-face, the sentiments of visual cues like smiling and frowning can be taken into account. In the article "Exploiting emoticons in sentiment analysis," by Hogenboom, the visual cues are replaced by the emoticons while communicating using the plain text. In order to function properly, machine learning polarity classifiers require lots of training. Often machine learning classifiers perform well when being trained in a particular domain, but in a different domain, the performance decreases. The lexicon based methods are more robust in performance across many domains and texts. This method uses more linguistic view on text data than abstracting from natural language by means of vectorization. Lexicon-based sentiment analysis contains compound, simple words and sentiments associated with the words. There are different tools to analyze the sentiments in the data. The binary representation of text indicating specific words that can convey a certain opinion is considered. The paper has focused on different vector representation, which has additional features that can distinguish between words. The other approach is lexicon analysis, which is more robust across the domains, but fails to harvest information from emoticons.

By the use of emoticons, users who communicate through computer-mediate have overcome the lack of personal contact. Emoticons have become the paralanguage of Web. By using the emoticons the user is able to indicate subtle change of mood to express stress, sarcasm, or sadness etc. even more than the face-to-face communication can express. By using of an emoticon in a sentence can change the sentiment of what the user wants to convey, either positively or negatively. For example, the movie is bad (☹) here not just the emoticon is considered but also analyzed how the emotion affects the text [15].

This article suggests a more effective framework where the emoticons can be analyzed in terms of sentiment analysis. The goal of the framework is to automate
sentiment analysis and determine the sentiment, which considers the information conveyed by the emoticons.

6. Methodology and Tools

6.1 Proposed Technique – Lexicon based sentiment analysis

"Sentiment analysis is the task of finding opinions of authors about specific entities [20]." Sentiment analysis is one of the newly evolving areas of research in computer science as now-a-days the decision making process depends on the opinions of others. One example of this is how the reviews written online affect the decision of the buyer during online shopping [20]. A sentence can be of two types: objective, which states the facts about the information, or subjective which contains opinions, beliefs, and views about something. Sentiment analysis is focussed on subjective information, analyzing if the opinion is positive, negative or neutral. There are various techniques of subjective sentiment analysis such as document level, sentence level, aspect-based, comparative and sentiment lexicon acquisition. In document level sentiment analysis, it is assumed that the document has opinions regarding something and then is analyzed using supervised or unsupervised sentiment analysis. Supervised analysis has a finite set of classes defined while in unsupervised analysis, the semantic orientation (degree or strength of the opinion) is determined. In order to have details of the opinions expressed in the multiple sentences of documents about the same entity, sentence level analysis is introduced. Two assumptions are made to analyse the sentence: the identity of the entity is known and the sentence has a single opinion, which makes it easier to split the sentence into phrases. The limitations of both of these techniques are that they work well only when a single entity is defined. It is challenging to detect sarcasm in both of them. The next technique is aspect-based analysis; opinions can be different for different attributes of an entity which cannot be analyzed as whole. This approach extracts the noun phrases and reduces the noise from them to analyze the opinion. One other method is comparative analysis, where the adjectives and adverbs more, less, most, least are used to analyze.

Sentiment lexicon acquisition will be the approach that we will use for our study, combined with aspect-based and comparative analysis. This method has three
different approaches: manual approach- hand coded lexicon, which is not feasible as it is too laborious to work, dictionary based approach- expansion of the words using available resources and the corpus based approach- identifying the adjectives that have consistent polarity. This technique is most feasible among all the techniques we studied because it is easy to write an algorithm using the corpus and the available resources with a high polarity of getting to the conclusion, whether the opinion is positive, negative or neutral. This approach is highly preferable as it calculates the semantic orientation automatically for the corpus. The semantic orientation tells the strength or the polarity of the opinion is inclined to which direction and how strong is it. As lexicon based analysis incorporates both the structural and semantic aspect of the content, it is more feasible to use. In usual sentiment analysis techniques, emoticons are not used as they are not the part of the text. We will also look into how exploiting emoticons can lead to achieve the polarity of the data and how do they affect the text.

6.2 Survey of Tools for analyzing the existing data

We have researched a few tools which we can possibly use to analyze the data we will collect from the application Geotagger, which are rapidminer, weka, and semantria. Rapidminer provides an integrated platform for data mining and analysis. Some of the advantages of the tool are it does not require coding knowledge, the GUI is very user friendly, it is an open source tool and also extensible, it allows the user to work on large data sets and runs on all platforms and OS [16]. Although rapidminer has many advantages, we didn’t find it to be feasible due to a few things. It works slowly with large data sets, errors are not clearly stated to understand and correct; and too many loops make the process complicated. The major issue we faced during the use of this software was there are very limited resources available to understand the data mining process and hence it is very time consuming to work with rapidminer. After using rapidminer, we tried to use the tool weka. We slightly preferred weka over rapidminer as it has a very good GUI, which is easy to understand and there are more resources available to understand the tool and use it. Weka also covers many data mining techniques like data preprocessing, clustering, analyzing etc. It is also an open source and portable, hence can be used on any platform as it is developed in Java [17]. Although weka was easy to use and a portable tool, it was mostly being used for clustering and data preprocessing, while we were looking into something more
specifically related to sentiment analysis, like a text analysis tool. After researching on text analytics and the tools available, we found out about Semantria. Semantria is a sentiment analysis tool owned by Lexalytics. It’s an excel plug-in which can be used for text and sentiment analytics and also incorporates knowledge based learning. It offers, in nine different languages, various methods of analysis like categorization, summarization, theme detection, sentiment analysis, etc. Our main focus will be doing sentiment analysis of the data we have collected using Geotagger [24].

6.3 Geotagger data analysis using sentiment analysis tool Semantria

The tool we used to analyze the data that we collected from geo-tagger is Semantria. It is a sentiment analysis tool owned by Lexalytics. It is an excel plug-in which can be used for text and sentiment analytics and also incorporates knowledge based learning. Semantria offers, in nine different languages, various methods of analysis like categorization, summarization, theme detection, sentiment analysis, etc. Our main focus will be doing sentiment analysis of the data we have. As semantria is an excel plug-in it is effortless to use and simple to analyze. After installing semantria, it has its own tab in excel; by clicking on the semantria tab the tool box will pop up. For fundamental analysis of the data, we used the detailed mode of the two modes that semantria offers. In detailed mode, it will analyze each review and select the phrases from that. Then phrases are assigned a positive or a negative polarity and depending on the number of phrases and their polarity semantria decides if the review is positive or negative. This method is lexical analysis. In the detailed mode, semantria will give you a document sentiment score after the analysis for each review; this is the sentiment for the block of text selected, that is determined by the phrases, which have their own sentiment score. During the analysis of the document, semantria uses three different NLP algorithms; Tokenization, Tagging and Chunking.

Tokenization is breaking up the review into individual tokens which can be words, phrases, punctuations, etc. It chops the sentence into pieces of words or terms removing the punctuations called tokens, which can be merged together and used for semantic processing [21]. Tokenization is followed by recognizing the part of speech for the tokens and labelling them respectively, which is known as Tagging. During the process of tagging, the tokens are recognized for what part of speech they are noun, verb, adverb, conjunction, etc. and respective labels are attached to the tokens [22].
Chunking is the next NLP process in the pipeline after tagging. The tokens are grouped together in grammatical chunks to extract information from data [23][24]. Semantria also gives the phrases in the output with the intensifier/negator if used for analyzing the data. For example, if ‘nice’ is a phrase for a positive score then ‘very’ can be an intensifier to increase the positive score while ‘not’ can be a negator to decrease the score.

We performed the sentiment analysis using the detailed mode of semantria on the Geotagger data we had collected. There were around 243 comments which were analyzed individually. The detailed sentiment analysis gave the document a score of either positive/negative/neutral, along with the phrases, intensifiers and negators which made the data easy to understand. Semantria, being an excel plug-in also gives an option to draw pivot charts to visualize the data easily. After the completion of the analysis, we plugged the data into a bar chart to check the sentiment of the complete document. In a total of 243 comments, there were 51 positive comments, 38 negative comments and 154 neutral comments. Figure 1 shows the data in the form of a chart; the green bar shows the positive comments, the red bar is for negative comments, and the grey bar shows the neutral comments. As seen in the chart, most of the comments are neutral; hence, the document has a neutral semantic score.
7. Geotagger formative participatory design session with some direct feedback with regards to different possible user interfaces

7.1 Participatory design session setup

An experiment was conducted with Geotagger Kidsteam to find an answer to the question whether sentiment analysis module is essential in the system and what do the users feel about being able to rate a tag. A survey was conducted in the kids team giving them various examples of the sentiment analysis module in the form of mock-ups. In addition to the survey, there was an open designing session where the kids can give their suggestions for the available designs or design their own prototypes.

7.2 Participatory design session conditions

The experimental setup consisted of three different situations, first was showing the kids team few example prototype and making them aware of the sentiment analysis module which we are trying to add. Secondly, a short survey was conducted asking them questions about their opinion regarding the prototypes. And then the kids team was divided into groups, each group consisting of few kids with one adult to design their own prototypes using the examples of the mock-ups.

7.3 Participants

The experiment was conducted at the research lab in the computer science department at Montclair State University. There were 12 participants including the kids team, college students and a Professor. The participants were from different age groups, 7 kids were 6 – 11 years old, 4 college students 17 – 18 years old and a Professor 38 years old. The kids team had 5 girls and 2 boys. In total there were 9 girls and 3 boys.

7.4 Methodology

The kids were asked to form a circle for the initial explanation and survey part. The experiment started with an informal introduction where the kids enjoyed sharing their name, age, how long they are with the Kidsteam and their favourite holiday treat. After the introduction, the kids were introduced to this new module of rating and asked if they remember the Geotagger app. The kids were shown the prototypes and
this was an open session where they could ask if they had any questions about the Geotagger app and the new prototypes. There were four different paper prototypes which showed the mock-up of the app and how it would look. It was very self-explanatory and easy to understand for the kids. Then a short 10 question survey was distributed. The kids were given 20–25 mins to fill out the survey. The adults were helping the kids if they had any doubts during the survey. After the survey, the team was divided into groups for the designing to prototype. Each group consisted of one or two kids with an adult. The whole team was divided into 4 groups. The groups were given paper, coloured pens and the prototypes for reference to build their own. The groups were given around 20–30 mins to discuss and design their own prototypes. After the designing session the groups shared their ideas where each member of the group had a chance to showcase their prototypes and give suggestions. Notes were taken as the kids were giving suggestions. The prototypes and the survey were collected at the end of the experiment.

7.5 Measures

Four different mock-ups were used as examples first one had happy and sad smiley faces by which the user could rate the tag, second was thumbs up/down button to express like or dislike, third one had radio buttons with statements to judge the enjoyment of the user and last one consisted on a combo-box with options yes/no/maybe. A questionnaire was used to measure the necessity of the new rating module in the Geotagger application. The questions were of different types like selecting one mock-up, rating the mock-ups and ranking them, what do you like the most/least, one yes/no/maybe question. Another measure was giving the participants an opportunity to design their own ideas and present their suggestions to everyone else. Both the measures were easy to understand and valuable for both the participants and us.

7.6 Results

In the first part which was the survey; the results of the questionnaire were analyzed in accordance to the rating and the ranks we collected. We also collected the things that the participants liked and did not like about the respective prototype. Most of the participants liked the thumbs up/down prototype. Seven participants liked the
thumbs up/down prototype, three participants chose the radio button one and rest of the two prototypes were selected by two participants each.

Out of 7 kids from which 4 girls and 1 boy, total 5 kids liked the thumbs up/down prototype, one boy liked the combo box prototype and one girl selected the radio button prototype. Amongst the 5 adults 2 of them liked the thumbs up/down prototype, 2 liked the radio button prototype and there was one vote for the smiley faces prototype.

The team was asked to rate and rank all the four prototypes and the average results of the rating and ranking are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Smiley faces prototype</th>
<th>Thumbs up/down prototype</th>
<th>Radio button prototype</th>
<th>Combo box prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average Rating</strong></td>
<td>3.27</td>
<td>4</td>
<td>2.67</td>
<td>3</td>
</tr>
<tr>
<td><strong>Average Rank</strong></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: Average Rating and Rank of the Prototypes

According to table 1, the most rated and ranked prototype was the Thumbs up/down prototype then the smiley face prototype. There was not a significant difference in the rating of the last two prototypes but the radio button one was ranked third while the combo box was fourth.

A question was asked about “Would you like to be able to rate the app?” All the adults recommended the module to be added in the Geotagger app. While in the kids team 3 kids recommended, 3 selected maybe and one did not recommend the module.

There were not many suggestions given by the participants but a few of them were as follows:

- The faces should be determined whether the tag has a lot of likes or dislikes
- An option of how to rate, comment/suggestion box, add on/edit to tags, a option between rating
7.7 Participatory design session questionnaire and suggestions / comments from the users

1) Which design do you like better amongst the four?

- Like/Dislike
- Like/Dislike
- Radio Button
- Like/Dislike
- Like/Dislike
- Like/Dislike
- Like/Dislike
- Radio Button
- Smiley Faces
- Combo box
- Radio Button
- Like/Dislike

2) How will you rate each one of them on a scale of 1 – 5? Considering 5 as highest.
3) What do you like about being able to rate tags?
   - It shows how much you would use them or like them, you can express your likes
   - Cool way to express feelings through likes and dislikes
   - That you can choose
   - You can see if people like your tags
   - It’s good
   - I think it might be a good idea you can give ideas
   - I like I can give my opinion
   - It’s an easier way to rate my satisfaction
   - I can say I like them
• I like that the tags are nice
• It lets people know what interest others
• It enhances the using experience and gives it a more social media feel

4) What do you NOT like about being able to rate tags?

• Nothing
• Too much likes not enough dislikes
• Nothing
• I don’t like the third picture
• Nothing
• You may hurt someone’s feelings
• I think it can hurt feeling and feel it’s a waste of time
• That if I don’t know
• Another thing to do, better be easy
• They don’t have colour
• The dislike button or “Bad Ratings” may give people feeling hesitant about posting

5) Rank the four demos in the order of preference, 1 being the highest.

<table>
<thead>
<tr>
<th>Prototypes</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

**Avg. Rank** 2.27 1.36 2.81 3.36

Table 3: Average Rank of the Prototypes

6) **What do you like the most? Why?**
   - I can like it and be done
   - I like the cool ideas like smileys and thumbs-up
   - I like the third one because you have sentences
   - I like the Like/Dislike idea because it is very simple
   - Easy click, Thumbs up
   - Expressing your idea, help others make it better
   - I like it is simple
   - Pictures
   - The pictures because they are sensible
   - Each rating is different but I like descriptions to be more detailed
   - The like + dislike button it’s not complicated and has a social feel

7) **What do you like the least? Why?**
   - You might have to explain the Maybe
   - Not much colour because colour is creative
   - I don’t like the third idea because you could already comment on the tags and you could say whatever you want
   - You cannot like the person and be mean on purpose
   - It doesn’t look good
   - Words
   - The word Geotagger because it is plane
   - The like and dislike button because it looks too much like Facebook
   - The “I think” written opinions are usually for connect sections
8) Would you like to be able to rate tags in the app?

<table>
<thead>
<tr>
<th>No</th>
<th>Maybe</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Yes</td>
<td>• Yes</td>
<td></td>
</tr>
<tr>
<td>• Yes</td>
<td>• Yes</td>
<td></td>
</tr>
<tr>
<td>• Yes</td>
<td>• No</td>
<td></td>
</tr>
<tr>
<td>• Maybe</td>
<td>• Maybe</td>
<td></td>
</tr>
<tr>
<td>• Maybe</td>
<td>• Yes</td>
<td></td>
</tr>
<tr>
<td>• Yes</td>
<td>• Maybe</td>
<td></td>
</tr>
<tr>
<td>• Yes</td>
<td>• Yes</td>
<td></td>
</tr>
</tbody>
</table>

9) What other comments or suggestions do you have?

- Write if you have to explain the Maybe
- None
- None
- The faces should be determined whether the tag has a lot of likes or dislikes
- None
- An option of how to rate, comment/suggestion box, add on/edit to tags, an option between rating
- None
- Change the labels of the smileys
- None
- None
- None
- N/A
7.8 Participatory design session user's suggested prototypes and overall suggestions.

In the second part of the participatory session, the participants were asked to design their own prototype in groups using the examples provided as reference. We collected different suggestions from the four different groups.

The first group used the thumbs up/down prototype as their reference and designed a new prototype in which in the bottom of the tag there should be a rating button to click and after that depending on the profile whether it is a girl or a boy the like/dislike symbol should be changed from thumbs up/down to something that girls like for girls and vice versa.

![Figure 2: Group 1 (Suggested Prototype)](image)

The second group designed a prototype using the smiley faces and the thumbs up/down prototypes as reference. They suggested to change the smiley faces according to the number of likes and dislikes the tag gets. List of the tags shows the smiley face and when a tag is clicked it will pop up an individual tag show the likes/dislikes with a share tag and comments option. This group also suggested that there should be auto correct for bad comments and the owner should be able block comments with a timeframe how long it should be blocked. Adding emojis near the comment box was an addition.
Third group combined the smiley face with the thumbs up/down and created a new icon. They also added the user should be given options for rating like emojis, like/dislikes, etc and let the user pick how he/she wants to rate the tag.

The fourth group blended the radio button and the combo box prototype together and came up with a mock up in which the good comments should be in green while the bad ones should be in red and after that adding a combo box asking the user for the recommendation with options yes/no/maybe.
8. IMPLEMENTATION

8.1 System Design

The results of the rating and the ranking section of the participatory design session indicated a preference towards the "like/dislike" prototype, followed by the "emoticons" prototype. During the open design session, the four groups of users proposed four different designs. Our observations were as follows; the first designed prototype considered the gender of the profile of the icons for the rating were anticipated. In the second design, the smiley faces were added based on the number of likes and dislikes. The third design recommended a new icon which was combined with thumbs up and a happy smiley and vice versa. While the fourth design was based on colour scheme, where good comments were green while bad comments were red. The common observation in all the four designs was that the users chose the emoticons as their reference to design their prototypes. We chose to implement the emoticons in the Geotagger app even though that was not the users' initial choice in the survey session. The "like/dislike" prototype was the highest rated prototype; however, emoticons are considered to be the pictorial representation of feelings or sentiments. Emoticons are considered to be a combination of 'emotions' and 'icons' which play a significant role in understanding sentiment in different scenarios [29], for example, communication in between two people, expressing views about a thing.
etc. They allow the users to express their feelings, and save time in understanding the graphical user interface (GUI) as compared to text. Considering a situation to collect user's feelings, users react to the emoticons like they would to humans which make the GUI more efficient. As Geotagger is an app mostly for kids, emoticons can make the GUI more user friendly, competent and correlate with the real world [25][26].

In the article, "Emoticons Convey Emotions without Cognition of Faces: An fMRI Study" the author shows through his study of Functional magnetic resonance imaging (fMRI – the technology that measures the activity of the brain by detecting changes associated with the blood flow [30]), that brain activities are associated with emoticons. This study also promotes the understanding of how emoticons enhance and affect the individual's behaviour [31]. Emoticons reduce the difficulty in expressing feelings using words and hence speed up communication. They make the interaction easier, more fun and interactive. Many users also apply emoticons sarcastically, which is usually possible only when there are more options and the GUI is extensible. Also many emoticons are aesthetically pleasant and look amusing, which might motivate the user to participate in the activity. They allow the users to convey more in less time [27].

The “like and dislike” button is not very flexible and versatile, and can possibly restrict the user from expressing his thoughts clearly. This can be illustrated by the latest phenomena that happened. One the most popular social media tools, Facebook, launched “Reactions” – an extended version to their like button. It was on user’s demand that Facebook went beyond the ubiquitous thumbs up button. Acknowledging that the “like” button is not the right sentiment for every occasion, the giant social network is offering “Reactions” now with six different emoticons being used. This is an initiative to offer the users a fast and simple way to express their thoughts or feelings in a positive and supportive way. "Not every moment you want to share is happy," CEO Mark Zuckerberg wrote in a Facebook post. "Sometimes you want to share something sad or frustrating. Our community has been asking for a dislike button for years, but not because people want to tell friends they don't like their posts. People want to express empathy and make it comfortable to share a wider range of emotions." Words of Mr. Zuckerberg explain that only “like and dislike” buttons do not solve the purpose when the users would like to express different feelings like
empathy, compassion, anger, etc. [28]. Emoticons are extensible and give more granularity to the user to express his/her sentiments clearly.

In considering the results from the Kidsteam feedback, although the users in the formative session rated the “like and dislike” prototype the highest, their suggestions on the open design session and the comments in the survey incline towards the “smiley” prototype. The “smiley” prototype was their second favourite and offered the ability to have more granularity of recording emotions and is a model that can also be scaled as more different sentiments need to be recorded. Hence, taking into account the purpose of the sentiment analysis module in the Geotagger project, we recommend the smiley face prototype to be added to the Geotagger app.

8.2 System Implementation

The principle objective of this study is to explore an area of sentiment analysis on the collected heterogeneous data in the form of text, images, and location details collected using the Geotagger application. Then observe and infer users’ reactions and opinions of the geographical data they collect. The sentiment analysis of the previously collected data gives an idea about the necessity to implement the mobile solution for rating purpose. The module with emoticons in the mobile app and the rating module on the website make it quicker for the user to express his/her opinion about a certain tag. The mobile solution for gathering the rating gives the user a better perspective to express his/her feelings using the emoticons. As the Geotagger application is mainly intended for children in helping them to improve their technical, social and cognitive skills, adding emoticons gives them a visually clear idea about being able to express their opinion in a fun way.

Considering the recommended prototype and all other aspects of using emoticons we implemented a design for rating a certain tag using emoticons. In this module, when the user edits a certain tag he/she is allowed to rate that tag. A question is asked, ‘How do you feel about this tag?’ with three smiley faces to answer the question. The three smileys are happy, sad, and normal emoticons which give the user a clear idea about his/her feelings at the first glance. Figure 6 shows the implementation of the sentiment analysis module in Geotagger.
9. Future Scope

In this thesis, we studied the sustainability science projects escalating towards the impact of Big Data and Sentiment Analysis on the sustainability projects. We demonstrated the need of sentiment analysis by analyzing the current data and drawing meaningful conclusions from it. The implementation of the sentiment analysis module in the existing Geotagger project, we proposed the need of understanding user’s emotions and the analysis of the comments can give us valuable information for advancement of the project. Looking towards the future we are trying to permit all the viewers to view other individual’s tag and rate it. This will allow us to get additional information regarding the tag beyond the valuable information of just the initial creator of the tag. Moreover, there can be some features added to the existing module like adding the emoticon on the tag list fragment which will be easily identified by the user. Considering the suggestions of the users in our participatory study, the comments can be analyzed and a red mark can identify a bad comment while a green mark can identify a good comment. With the increasing amount of data in the Geotagger application, it can face a Big Data problem in the future; taking steps towards the area of sentiment and data analysis can help to resolve the issue and make the most out of the information for the advancement in the area of sustainability science.
10. CONCLUSION

In this thesis, we presented our work in the field of data mining and predictive analysis. We originally started our work with sustainability sciences and citizen sciences while working on Geotagger, platform for environmental enquiry. Further we expanded looking into how sentiment analysis can help to analyze the sustainability issues. We have done a survey in different areas like opinion mining, sustainability science and sentiment analysis techniques. We analyzed the data recorded by the Geotagger application using Semantria, followed by a participatory study to analyze the need of sentiment analysis module for Geotagger. We concluded that it was necessary to add a rating module to the app to better understand and user's sentiments about a specific tag. We implemented the emoticons prototype in the existing app, helping the users to improve their technical, social and cognitive skills. Whereas, adding emoticons gives the users a visually clear idea about being able to express their opinions, which can help in better decision making.
Bibliography


25. Emoticons, [Online], Available:
   http://mason.gmu.edu/~afinn/html/tele/conventions/emoticons.htm

26. C. Seiter., (2015), Reasons to use emoticons in your writing and social media, according to science, [Online], Available: https://blog.bufferapp.com/7-reasons-use-emoticons-writing-social-media-according-science


30. Functional magnetic resonance imaging. [Online], Available: https://en.wikipedia.org/wiki/Functional_magnetic_resonance_imaging

Appendices

Code

Tag Edit
package com.hci.geotagger.activities;
import java.io.File;
import java.text.DecimalFormat;
import android.app.ActionBar;
import android.app.FragmentManager;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.provider.Settings;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.graphics.Bitmap;
import com.hci.geotagger.GeotaggerApplication;
import com.hci.geotagger.R;
import com.hci.geotagger.activities.common.BaseActivity;
import com.hci.geotagger.cache.CacheDatabase;
import com.hci.geotagger.common.Constants;
import com.hci.geotagger.dbhandler.DbHandlerConstants;
import com.hci.geotagger.dbhandler.DbHandlerDbResponse;
import com.hci.geotagger.dbhandler.DbHandlerDbResponse.DbMessageResponseInterface;
import com.hci.geotagger.objects.Adventure;
import com.hci.geotagger.objects.AdventureTags;
import com.hci.geotagger.objects.GeoLocation;
import com.hci.geotagger.objects.Tag;
/**
 * Add Tag activity allows the user to add new tags to the database.
 * This activity also supports Edit Tag capability.
 * This includes setting fields such as name/description and also
 * setting an image for the tag and using geo-location.
 * <p/>
 * TODO: For edit should check if fields are changed, if not then keep the Save button disabled
 * TODO: Add ability to change the Tag picture
 */

public class AddTagActivity extends BaseActivity implements DbMessageResponseInterface {
    private String TAG = "AddTagActivity";
    private final int CONTEXT_DELETE_ID = 1;
    final Context c = AddTagActivity.this;
    private Adventure adventure;
    Button btnOk;
    EditText txtName, txtDesc, txtLoc;
    TextView txtOrigLoc;
    private CheckBox chkGPS;
    private CheckBox chkOrigLocation;
    private TextView locationGPS;
    private TextView labelGPS;
    String gpsUnknown;
    String gpsNoSignal;
    AddImageFragment imageFragment;
    private DecimalFormat lldf = new DecimalFormat("#.000000");
    private LocationListener listener;
    private boolean gpsEnabled = false;
    private boolean networkEnabled = false;
    private LocationManager lm;
    private Location location;
    private int intentFlags = 0;
    private static DbHandlerResponse rspHandler;
    private GeotaggerApplication app;
    private Tag tag;
    private Long savedTagID = 0L;
    private boolean updateTag = false;
    private int getScaledImageID;
    private int getTagAfterAdd = 0;
    private int getTagInitial = 0;
    private int iRatingScore = 3;
    ImageView imgvw_smile1, imgvw_smile2, imgvw_smile3;
    LinearLayout linlay_smile1, linlay_smile2, linlay_smile3;
    String iratingScore = "3";
/**
 * Initializes GPS location listener in case user uses gps coordinates for
 * added tag, a new ImageHandler which will retrieve image from mediastore(using
 * generic method regardless of device type), scale it
 * to avoid using too much memory(which may cause crash on older phone) and
 * upload it to server, and of initialize ui elements.
 */

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.tfcfrv//y_tf<7<7_ta< t
ActionBar ab = getActionBar();

//buttons
btnOk = (Button) findViewById(R.id.addtag btnOk);

Intent intent = getIntent();
Bundle bundle = intent.getExtras();

intentFlags = intent.getFlags();
if (bundle.containsKey(Constants.EXTRA_ADVENTURE)) {
    ab.setTitle(R.string.add_tag);
    adventure = (Adventure)
    bundle.getSerializable( Constants.EXTRA_ADVENTURE);
    updateTag = false;
} else if (bundle.containsKey( Constants.EXTRA_TAGID)) {
    ab.setTitle(R.string.edit_tag);
    btnOk.setText(R.string.save);
    setupProgress(getString(R.string.progress_loading));
    showProgress();
    savedTagID = bundle.getLong( Constants.EXTRA_TAGID);
    tag = new Tag(savedTagID);
    updateTag = true;
}

//text fields
txtName = (EditText) findViewById(R.id.addtag_name);
txtDesc = (EditText) findViewById(R.id.addtag_desc);
txtLoc = (EditText) findViewById(R.id.addtag_location);
txtOrigLoc = (TextView) findViewById(R.id.origlocationvalue);

//Smiles
imgvw_smile1 = (ImageView) findViewById(R.id.imgvw_smile1);
imgvw_smile2 = (ImageView) findViewById(R.id.imgvw_smile2);
imgvw_smile3 = (ImageView) findViewById(R.id.imgvw_smile3);

//Linerlay
linlay_smile1 = (LinearLayout) findViewById(R.id.linlay_smile1);
linlay_smile2 = (LinearLayout) findViewById(R.id.linlay_smile2);
linlay_smile3 = (LinearLayout) findViewById(R.id.linlay_smile3);

//
txtName.addTextChangedListener(setButtonStateWatcher);
txtDesc.addTextChangedListener(setButtonStateWatcher);

// Original Location checkbox (if edit)
chkOrigLocation = (CheckBox) findViewById(R.id.origlocationcheckbox);
//Check box
chkGPS = (CheckBox) findViewById(R.id.addtag_useGPS);
// Set disabled until a valid location is received
chkGPS.setEnabled(false);
locationGPS = (TextView) findViewById(R.id.locateTextView);
gpsUnknown = getString(R.string.gps_location_unknown);
gpsNoSignal = getString(R.string.gps_no_signal);
locationGPS.setText(gpsUnknown);

labelGPS = (TextView) findViewById(R.id.addtag_lblLoc);

labelGPS.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
    }
});

FragmentManager fm = getFragmentManager();
imageFragment = (AddImageFragment) fm.findFragmentByTag(R.id.addcomment_image_fragment);

//initialize location components
lm = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
location = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
gpsEnabled = lm.isProviderEnabled(LocationManager.GPS_PROVIDER);
networkEnabled = lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

listener = new LocationListener() //initialization of LocationListener
{
    @Override
    public void onLocationChanged(Location locat) //update location when it changes
    {
        location = locat;
        Log.d(TAG, "onLocationChanged called");
        String locationText = "" + lat + "", " + lldf.format(locat.getLongitude());
        locationGPS.setText(locationText);
        chkGPS.setEnabled(true);
    }

    @Override
    public void onProviderDisabled(String provider) {
        // TODO Auto-generated method stub
        Log.d(TAG, "onProviderDisabled called");
        if (location == null) {
            locationGPS.setText(gpsNoSignal);
            chkGPS.setEnabled(false);
        }
    }
}
```java
@Override
public void onProviderEnabled(String provider) {
    // TODO Auto-generated method stub
    Log.d(TAG, "onProviderEnabled called");
    if (location == null)
        locationGPS.setText(gpsUnknown);
    else
        chkGPS.setEnabled(true);
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
    // TODO Auto-generated method stub
    Log.d(TAG, "onStatusChanged called");
}

lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000L, 500.0f, listener);

// If the orientation was changed, reload the image
if (savedInstanceState != null) {
    String savedUri = (savedInstanceState.getString("imageUri"));
    if (savedUri != null) {
        Uri imgUri = Uri.parse(savedUri);
        if (!imageFragment.setImage(imgUri))
            Toast.makeText(c, getString(R.string.toast_problem_loadingimage), Toast.LENGTH_SHORT).show();
    } else {
        imageFragment.clearImage();
    }
}

// Smile btn click
imgvw_smile1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        ratingScore = "1";
        linlay_smile1.setBackgroundColor(Color.TRANSPARENT);
        linlay_smile2.setBackgroundColor(Color.TRANSPARENT);
    }
});

imgvw_smile2.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        ratingScore = "2";
        linlay_smile1.setBackgroundColor(Color.TRANSPARENT);
        linlay_smile2.setBackgroundColor(Color.TRANSPARENT);
    }
});
```
imgvw_smile3.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        iratingScore = "3";
        linlay_smile1.setBackgroundColor(Color.TRANSPARENT);
        linlay_smile2.setBackgroundColor(Color.TRANSPARENT);
        linlay_smile3.setBackgroundResource(R.drawable.tf_c);
    }
});

// Add button action
btnOk.setOnClickListener(new OnClickListener() {
    public void onClick(View view) {
        String name = txtName.getText().toString();
        Toast.makeText(AddTagActivity.this, "Tag added-Loaded successfully", Toast.LENGTH_SHORT).show();
        // Make sure the required fields are set
        if (name.isEmpty()) {
            Toast.makeText(c, R.string.problem_tagneedsname, Toast.LENGTH_SHORT).show();
            btnOk.setEnabled(true);
        } else {
            // Create a basic Tag object, which will be updated/filled
            tag = new Tag(savedTagID);
            tag.setName(name);
            String description = txtDesc.getText().toString();
            tag.setDescription(description);
            tag.setLocationString(txtLoc.getText().toString());
            tag.setRatingScore(Integer.parseInt(iratingScore.toString()));
            GeoLocation geo = new GeoLocation(0, 0);
            //if the user wants to use GPS coordinates, get the current location to store in tag
            if (chkGPS.isChecked()) {
                if (location == null) {
                    Toast.makeText(c, R.string.toast_acquiringsignal, Toast.LENGTH_SHORT).show();
                    //TODO: This causes a crash!!!
                    onClick(view); //recursive call if location cannot be found
                } else {
                    geo.setLatitude(location.getLatitude());
                    geo.setLongitude(location.getLongitude());
                }
            } else if (updateTag && chkOrigLocation.isChecked()) {
                geo = tag.getLocation();
            }
        }
    }
});
tag.setLocation(geo);

// attempt to add tag to db
startAddingTag();
CacheDatabase cacheDatabase = new CacheDatabase(AddTagActivity.this);
cacheDatabase.addMYTag(tag);
if (intentFlags == 2) {
    adventure.addStoreTagList(tag);
}
}
app = (GeotaggerApplication) getApplication();
rspHandler = new DbHandlerResponse(TAG, this, this);
app.addResponseHandler(rspHandler);

LinearLayout origLayout = (LinearLayout) findViewById(R.id.origlocationlayout);
LinearLayout origBorder = (LinearLayout) findViewById(R.id.origlocationborder);
if (updateTag) {
    btnOk.setEnabled(true);
    getTaglnitial = app.sendMsgToDbHandler(rspHandler, this, DbHandlerConstants.DBMSG_GET_TAG, tag);
    origLayout.setVisibility(View.VISIBLE);
    origBorder.setVisibihty(View.VISIBLE);
    chkOrigLocation.setChecked(true);
    chkOrigLocation.setOnCheckedChangeListener(new
        CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton argO, boolean state) {
                // Cannot have both checked
                if (chkGPS.isChecked())
                    chkGPS.setChecked(false);
            }
        });
    chkGPS.setOnCheckedChangeListener(new
        CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton argO, boolean state) {
                // Cannot have both checked
                if (chkOrigLocation.isChecked())
                    chkOrigLocation.setChecked(false);
            }
        });
    } else {
        btnOk.setEnabled(false);
        origLayout.setVisibility(View.GONE);
        origBorder.setVisibility(View.GONE);
    }
}
private TextWatcher setButtonStateWatcher = new TextWatcher() {
    public void afterTextChanged(Editable s) {
        if (txtName.getText().length() > 0 && txtDesc.getText().length() > 0) {
            btnOk.setEnabled(true);
        } else {
            btnOk.setEnabled(false);
        }
    }
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }
    public void onTextChanged(CharSequence s, int start, int before, int count) {
    }
};

private void updateContentsQ {

    // try {
    String rating = tag.getDescription().substring(tag.getDescription().length() - 1);
    if (rating.contains("1") || rating.contains("2") || rating.contains("3")) {
        int irating = 3;
        irating = Integer.parseInt(rating.toString());
        // tag.setRatingScore(irating);
        // tags.get(i).setRatingScore(Integer.parseInt(tags.get(i).getDescription().substring(tags.get(i).getDescription().length() - 1), 3));
        tag.setDescription(tag.getDescription().substring(0, tag.getDescription().length() - 1));
    } else {
        tag.setRatingScore(3);
    }

    // catch (Exception e) {
    //
    //}

    Bitmap bitmap;
    bitmap = tag.getBitmap();
    if (bitmap != null) {
        imageFragment.setImage(bitmap);
    } else {
        imageFragment.clearImage();
    }

    // text fields
    txtName.setText(tag.getName());
    txtLoc.setText(tag.getLocationString());

    // text fields
    txtName.setText(tag.getName());
    txtLoc.setText(tag.getLocationString());
/**
 * display sescription
 */
// String description = tag.getDescription();
// String strrating = description.substring(description.length() - 1);
// txtDesc.setText(description.substring(0, description.length() - 1));
// set smilies tag update
int iirating = tag.getRatingScore();
if (iirating == 1) {
    linlay_smile1.setBackgroundResource(R.drawable.a/?/?_c/rc/e);
    linlay_smile2.setBackgroundColor(Color.TRANSPARENT);
    linlay_smile3.setBackgroundColor(Color.TRANSPARENT);
} else if (iirating == 2) {
    linlay_smile1.setBackgroundColor(Color.TRANSPARENT);
    linlay_smile2.setBackgroundResource(R.drawable.£zpp_czrc/e);
    linlay_smile3.setBackgroundColor(Color.TRANSPARENT);
} else {
    linlay_smile1.setBackgroundColor(Color.TRANSPARENT);
    linlay_smile2.setBackgroudColor(Color.TRANSPARENT);
    linlay_smile3.setBackgroundResource(R.drawable.a/?/?_czVc/e);
}
GeoLocation geo = tag.getLocation();
if (geo.getLatitude() == 0.0 && geo.getLongitude() == 0.0) {
    if (updateTag) {
        chkOrigLocation.setChecked(false);
        chkOrigLocation.setEnabled(false);
    }
    txtOrigLoc.setText(this.getString(R.string.a£/<i_to<
o_orig_location))
} else {
    if (updateTag) {
        chkOrigLocation.setChecked(true);
        chkOrigLocation.setEnabled(true);
    }
    String locationText = lldf.format(geo.getLatitude()) + ", " +
    lldf.format(geo.getLongitude());
    txtOrigLoc.setText(locationText);
    chkGPS.setEnabled(false);
}
/**
 * This method will call the appropriate database action,
 * Add to create a new Tag and Update to edit an existing
 * tag
 */
private void startAddingTag() { 48
setupProgress(c.getResources().getString(R.string.progress_£taW)));
showProgress();

// Set the image associated with the tag, if there is one
if (this.updateTag) {
   // TODO: add update to the Image, if possible
   app.sendMsgToDbHandler(rspHandler, this,
   DbHandlerConstants.DBMSG_UPDATE, tag);
} else {
   tag.setImageUploadFile(imageFragment.getCurrentImage());
   if (imageFragment.getCurrentImage() != null) {
      app.sendMsgToDbHandler(rspHandler, this,
      DbHandlerConstants.DBMSG_UPLOAD_IMAGE, imageFragment.getCurrentImage());
   } else {
      app.sendMsgToDbHandler(rspHandler, this,
      DbHandlerConstants.DBMSG_ADD, tag);
   }
}

@Override
public void DbMessageResponse_DBCallback(int action, int msgID, boolean success,
boolean done, Object response) {
   Log.d(TAG, "Entered DBGetCallback");
   String msg;

   if (success) {
      switch (action) {
      case DbHandlerConstants.DBMSG_UPDATE:
         finishAddingTagO;
         break;
      case DbHandlerConstants.DBMSG_UPLOAD_IMAGE:
         Long imageID = (Long) response;
         tag.setImageld(imageID);
         app.sendMsgToDbHandler(rspHandler, this,
         DbHandlerConstants.DBMSG_ADD, tag);
         break;
      case DbHandlerConstants.DBMSG_ADD:
         if (response instanceof Tag) {
            tag = (Tag) response;
            // If adding the Tag to and adventure then create the relationship
            if (intentFlags == 1 & & adventure != null) {
               AdventureTags advTag = new AdventureTags(adventure.getld(),
               tag.getld());
               app.sendMsgToDbHandler(rspHandler, this,
               DbHandlerConstants.DBMSG_ADD, advTag);
            } else {
               getTagAfterAdd = app.sendMsgToDbHandler(rspHandler, this,
               DbHandlerConstants.DBMSG_GET_TAG, tag);
            }
            } else if (response instanceof AdventureTags) {
               getTagAfterAdd = app.sendMsgToDbHandler(rspHandler, this,
               DbHandlerConstants.DBMSG_GET_TAG, tag);
         }
      }
break;
case DbHandlerConstants.DBMSG_GET_TAG:
  if (msgID == getTagAfterAdd) {
    finishAddingTag();
  } else {
    tag = (Tag) response;
    String url = tag.getImageUrl();
    if (url != null && url.length() > 0) {
      DbHandlerScaledImageReq gsi = new DbHandlerScaledImageReq();
      gsi.width = (int)(getResources().getDimension(R.dimen.tag_image_max_size));
      gsi.height = (int)(getResources().getDimension(R.dimen.tag_image_max_size));
      gsi.urls = new String[1];
      gsi.urls[0] = url;
      getScaledImageID = app.sendMsgToDbHandler(rspHandler, this,
DbHandlerConstants.DBMSG_GET_SCALED_IMAGES, gsi);
    } else {
      updateContents();
      stopProgress();
    }
  }
  break;
case DbHandlerConstants.DBMSG_GET_SCALED_IMAGES:
  if (msgID == getScaledImageID) {
    if (success) {
      DbHandlerScaledImageRsp sir = (DbHandlerScaledImageRsp) response;
      tag.setBitmap(sir.bitmap);
    }
  } else {
    updateContents();
    stopProgress();
  }
  break;
} else {
  stopProgress();
}
switch (action) {
  case DbHandlerConstants.DBMSG_UPLOAD_IMAGE:
    msg = this.getResources().getString(R.string.toast_uploadimage_failure);
    Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
    break;
  case DbHandlerConstants.DBMSG_ADD:
    if (response instanceof AdventureTags) {
      msg = this.getResources().getString(R.string.toast_addadventuretag_failure);
    } else {
      msg = this.getResources().getString(R.string.toast_addadventuretag_failure);
    }
    Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
    break;
case DbHandlerConstants.DBMSG_GET_TAG:
    case DbHandlerConstants.DBMSG_GET_SCALED_IMAGES:
        updateContents();
        break;
    }
}

private void finishAddingTag() {
    String msg;
    if (updateTag)
        msg = getString(R.string.toast_edittag_success);
    else
        msg = getString(R.string.toast_addtag_success);
    Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
    stopProgress();
    setResult(RESULT_OK);
    finish();
}

@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    File curlImage = imageFragment.getCurrentImage();
    if (curlImage != null)
        outState.putString("imageUri", curlImage.toString());
}

/**
 * Defines the context menu for when an image view is long pressed
 */
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {
    if (v.getId() == R.id.addtag_imgView) {
        AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;
        menu.setHeaderTitle("Tag Image");
        menu.add(Menu.NONE, CONTEXT_DELETE_ID, Menu.NONE, "Clear");
    }
}

/**
 * Context handler for deleting an image on long press
 */
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        //if the user deletes the image, set the flag to false,
        //reset the imageview size and image to default
        case CONTEXT_DELETE_ID:
            if (imageFragment.getHasImage()) {
                imageFragment.clearImage();
            }
    }
Tag View

package com.hci.geotagger.activities;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import android.graphics.Color;
import android.util.Log;
import com.hci.geotagger.GeotaggerApplication;
import com.hci.geotagger.R;
import com.hci.geotagger.activities.CommentViewFragment.ICommentViewCallBack;
import com.hci.geotagger.activities.DescriptionViewFragment.IDescriptionViewCallback;
import com.hci.geotagger.activities.common.BaseActivity;
import com.hci.geotagger.cache.CacheDatabase;
import com.hci.geotagger.common.Constants;
import com.hci.geotagger.common.UserSession;
import com.hci.geotagger.dbhandler.DbHandlerScaledImageReq;
import com.hci.geotagger.dbhandler.DbHandlerConstants;
import com.hci.geotagger.dbhandler.DbHandlerResponse;
import com.hci.geotagger.dbhandler.DbHandlerResponse.DbMessageResponseInterface;
import com.hci.geotagger.dbhandler.DbHandlerScaledImageRsp;
import com.hci.geotagger.gui.MapViewHandler;
import com.hci.geotagger.gui.ScaleImageView;
import com.hci.geotagger.gui.ScaleImageView.ScaleImageViewCallbacks;
import com.hci.geotagger.objects.Adventure;
import com.hci.geotagger.objects.Comment;
import com.hci.geotagger.objects.GeoLocation;
import com.hci.geotagger.objects.Tag;
import android.os.Bundle;
import android.app.ActionBar;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.drawable.Drawable;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.RatingBar;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.FrameLayout;

/**
 * This class allows user to view all information associated with a given tag. This includes
 comments, location data, and time stamp. This class implements SensorEventListener
 although it does not use
 * its overridden methods(onAccuracyChanged and onSensorChanged) as they are not used in
 this iteration of geoTagger. Once the developer is sure, this implementation can be removed
 from this class. This
 * class also implements ICommentViewCallBack and IDescriptionViewCallback which are
 used to populate the DescriptionViewFragment and CommentViewFragment after they have
 been resolved. Note these
 * callbacks were deemed necessary since we must be sure the fragments were completely
 instantiated and rendered before populating them with downloaded information. Therefore,
 we must utilize
 * callbacks used in those fragments.
 */
// public class TagViewActivity extends Activity implements SensorEventListener

public class TagViewActivity extends BaseActivity
        implements SensorEventListener, ICommentViewCallBack, IDescriptionViewCallback,
DbMessageResponseInterface, ScaleImageCallbacks
{
    private static final String
            TAG
            = "TagViewActivity";

    public static final String
            TAG UPDATED = "Updated";
    public static final String
            TAG DELETED = "Deleted";
    public static final String
            EXTRA_TAGID = "TagID";

    ActionBar actionBar;
    TextView txtownerAndTime, txt tagLocation, txt tagDescription, txt Rating,
    txt currentLoc, txt distance, txt latLong;
    ImageView imgtaglmage, commentrowthumbnail, compassTriangle, imgvesmile;
    ImageView imgcommentlmage;
    ImageView btnRating;
    Button commentBtn, navBtn, handle, revealedHandle;
    RelativeLayout drawer;
    EditText commentTxt;
    Dialog ratingDialog;
    RatingBar ratingBar;
ListView commentList;
String url;

private Drawable icon;

private DecimalFormat ll df = new DecimalFormat("#.000000");

private long currentTagID = 0;
private Tag currentTag = null;

// fields needed for the location on tag
private GeoLocation geo;

private ArrayList<Comment> comments = null;

private HashMap<String, Bitmap> thumbCache;

/*
 * Below variables are for New MobSci UI
 */

private LinearLayout lowerArea;

private LinearLayout descriptionTabBtn; // press to view tag description
// Below comments variable renamed to commentsToggleBtn to prevent collision
// with comments above
private LinearLayout commentsTabBtn; // press to view tag comments
private LinearLayout mapViewTabBtn; // press to view tag's position on a map
private FrameLayout tagViewContent; // utilizes a frame layout to use
   // fragments
private LinearLayout mapViewContent;

// changed below because polymorphism didn't seem to work for some
// reason (only superclass methods visible)
private Fragment commentViewFragment = null; // fragment that contains the
   // comments for the tag
private Fragment descriptionViewFragment = null; // fragment that contains
   // the description for
   // the tag
private MapViewHandler mapViewHandler;
private FragmentManager fm; // FragmentManager that allows for switching
   // between fragments
private FragmentTransaction transaction; // used to switch between fragments

// New UI Elements
TextView descriptionTxtView;
TextView dateCreatedTxtView;
TextView timeStampTxtView;
TextView lblLocationTxtView;
TextView locationTxtView;
ScalelmageView tagPhotoImgView;

private static DbHandlerResponse rspHandler;
private GeotaggerApplication mApp = null;
private int removeCommentIndex;
private int commentlmageGetlD;
private int taglmageGetID;
private boolean commentslnitialized = false;

/**
 * Overridden onCreate method that instantiates UI elements, downloads associated tag information, and displays them to the UI once they are finished.
 */

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_tag_view);

    mApp = (GeotaggerApplication) getApplication();
    if (mApp != null)
    {
        rspHandler = new DbHandlerResponse(TAG, this, this);
        mApp.addResponseHandler(rspHandler);
    }
    Toast.makeText(this, "Data Loaded successfully demo act", Toast.LENGTH_SHORT).show();
    Intent intent = getIntent();
    Bundle bundle = intent.getExtras();
    currentTag = (Tag) bundle.getSerializable("selectedTag");
    // TODO: Need to determine if this Tag is owned by this user
    // if the current tag is owned by the logged in user, give option to
    // delete it
    if (currentTag.getOwnerId() == UserSession.CURRENT_USER.getldQ)
    {
        optionsMenuID = R.menu.view_tag_delete;
    }
    else
    {
        optionsMenuID = R.menu.view_tag;
    }
    imgve_smile = (ImageView) findViewById(R.id.imgve_smile);
    // optionsMenuID = R.menu.view_tag;

    // set up comments
    comments = new ArrayList<Comment>();
    thumbCache = new HashMap<String, Bitmap>();
    lowerArea = (LinearLayout) findViewById(R.id.lowerArea);
    descriptionTabBtn = (LinearLayout) findViewById(R.id.description_button); // buttons
    commentsTabBtn = (LinearLayout) findViewById(R.id.comments_button);
    mapViewTabBtn = (LinearLayout) findViewById(R.id.map_button);
    tagViewContent = (FrameLayout) findViewById(R.id.tagViewContent);
    mapViewContent = (LinearLayout) findViewById(R.id.mapViewContent);
mapViewContent.setVisibility(View.GONE);
tagViewContent.setVisibility(View.VISIBLE);

commentViewFragment = new CommentViewFragment(this);
descriptionViewFragment = new DescriptionViewFragment(this);

Fragment mapFragment = getFragmentManager().findFragmentById(R.id.map);
mapViewHandler = new MapViewHandler(mapFragment, this);

fm = getFragmentManager();

// show description tab by default
transaction = fm.beginTransaction();
transaction.replace(R.id.tagViewContent, descriptionViewFragment);
transaction.commit();

descriptionTabBtn.setEnabled(false); // do not allow multiple clicks causing // multiple transactions
Log.d(TAG, "Displaying new UI elements");

// implement the button listeners
descriptionTabBtn.setOnClickListener(new OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        if (!descriptionViewFragment.isAdded())
        {

            // show the description fragment
            transaction = fm.beginTransaction();
            transaction.replace(R.id.tagViewContent, descriptionViewFragment);

            transaction.commit();
        }
        else
        {
            transaction.show(descriptionViewFragment);
        }

        if (tagViewContent.getVisibility() != View.VISIBLE)
        {
            mapViewContent.setVisibility(View.GONE);
            tagViewContent.setVisibility(View.VISIBLE);
        }

        // disable the description button, re-enable the others
        descriptionTabBtn.setEnabled(false);
        commentsTabBtn.setEnabled(true);
        mapViewTabBtn.setEnabled(true);

        // displayTagInformation
        Log.d(TAG, "Description Toggled");
commentsTabBtn.setOnClickListener(new OnClickLisener()
{
    @Override
    public void onClick(View v)
    {
        Log.d(TAG, "Comments toggle button pressed");

        ((CommentViewFragment) commentViewFragment).setTagIndex(currentTag);

        if (!commentViewFragment.isAdded())
        {
            // show the description fragment
            transaction = fm.beginTransaction();
            transaction.replace(R.id.tagViewContent, commentViewFragment);
            transaction.commit();
        }
        else
        {
            transaction.show(commentViewFragment);
        }

        if (tagViewContent.getVisibility() != View.VISIBLE)
        {
            mapViewContent.setVisibility(View.GONE);
            tagViewContent.setVisibility(View.VISIBLE);
        }

        // disable the comments button, re-enable the others
        descriptionTabBtn.setEnabled(true);
        commentsTabBtn.setEnabled(false);
        mapViewTabBtn.setEnabled(true);
    }
});

mapViewTabBtn.setOnClickListener(new OnClickLisener()
{
    @Override
    public void onClick(View v)
    {
        Log.d(TAG, "Mapview toggle button pressed");

        if (mapViewContent.getVisibility() != View.VISIBLE)
        {
            tagViewContent.setVisibility(View.GONE);
            mapViewContent.setVisibility(View.VISIBLE);
        }

        mapViewHandler.addLocation(currentTag);

        // set location name

        mapViewContent.setVisibility(View.VISIBLE);
        tagViewContent.setVisibility(View.GONE);
        mapViewContent.setVisibility(View.VISIBLE);

        // disable the comments button, re-enable the others
        descriptionTabBtn.setEnabled(true);
        commentsTabBtn.setEnabled(false);
        mapViewTabBtn.setEnabled(true);
    }
});


```java
// mapViewHandler.setLocationName(currentTag.getName());

mapViewHandler.show();

// disable the map view button, re-enable the others
descriptionTabBtn.setEnabled(true);
commentsTabBtn.setEnabled(true);
mapViewTabBtn.setEnabled(false);
}
});

@Override
public void onBackPressed()
{
    if (lowerArea.getVisibility() == View.VISIBLE)
    {
        Intent returnIntent = new Intent();
        returnIntent.putExtra(TAG_UPDATED, currentTagID);
        setResult(RESULT_OK, returnIntent);
        finish();
    }
    else
    {
        lowerArea.setVisibility(View.VISIBLE);
    }
}

private void updateCommentFragment()
{
    commentsInitialized = true;
    ((CommentViewFragment) commentViewFragment).clearComments();
    ((CommentViewFragment) commentViewFragment).setCommentThumbCache(thumbCache);
    ((CommentViewFragment) commentViewFragment).notifyCommentAdapterDataChanged();
    for (int i = 0; i < comments.size(); i++)
    {
        ((CommentViewFragment) commentViewFragment).addComment(comments.get(i));
    }
    // commentList.setSelection(comments.size()-1);
    ((CommentViewFragment) commentViewFragment).setCommentListSelection(comments.size() - 1);
    ((CommentViewFragment) commentViewFragment).notifyCommentAdapterDataChanged();
}

@Override
public void DbMessageResponse_DBCallback(int action, int msgID, boolean success, boolean done, Object response)
{
    Log.d(TAG, "Entered DBGetCallback");
    switch (action)
    {
```
{  
case DbHandlerConstants.DBMSG_GET_TAG:
   // This is a response when we re-get the tag after an update
   // currentTag = (Tag)response;
   /*
   * String url = tag.getImageUrl(); if (url != null & url.length() > 0) {
   DbHandlerScaledImageReq gsi = new DbHandlerScaledImageReq();
   gsi.width = (int) getResources().getDimension(R.dimen.tag_image_max_size);
   gsi.height = (int) getResources().getDimension(R.dimen.tag_image_max_size);
   gsi.urls = new String[1];
   * gsi.urls[0] = url;
   *
   * getScaledImageID = app.sendMsgToDbHandler(rspHandler, this,
DbHandlerConstants.DBMSG_GET_SCALED.Images, gsi); } else {
   */
   displayTag();
   break;
}

case DbHandlerConstants.DBMSG_GET_TAG_COMMENTS:
   if (success)
   {
      comments = (ArrayList<Comment>) response;
      if (comments != null & comments.size() > 0)
      {
         DbHandlerScaledImageReq gsi = new DbHandlerScaledImageReq();
         gsi.width = (int) getResources().getDimension(R.dimen.thumbnail_width);
         gsi.height = (int) getResources().getDimension(R.dimen.thumbnail_height);

         int numImages = 0;
         // Calculate the number of comments that have images
         for (Comment c : comments)
         {
            String url = c.getImageURL();
            if (url != null & !url.equals("") & !thumbCache.containsKey(url))
               numImages++;
         }

         if (numImages > 0)
         {
            gsi.urls = new String[numImages];
            int curUrl = 0;
            // loop through tags and cache their images if they have them
            for (Comment c : comments)
            {
               String url = c.getImageURL();
               Log.d(TAG, "URL is " + url);
               // if tag has image url, download image and cache it
               if (url != null & !url.equals("") &
!thumbCache.containsKey(url))
               {
                  gsi.urls[curUrl++] = url;
               }
            }  
      }
   }
   commentImageGetID = mApp.sendMsgToDbHandler(rspHandler, }
DbHandlerConstants.DBMSG_GET_SCALED_IMAGES, gsi);
    return;
    }
    updateCommentFragment();
    }
  }
  stopProgress();
  break;
  case DbHandlerConstants.DBMSG_GET_SCALED_IMAGES:
    if (msgID == commentImageGetID)
    {
      if (success)
      {
        DbHandlerScaledImageRsp sir = (DbHandlerScaledImageRsp) response;
        thumbCache.put(sir.url, sir.bitmap);
      }
      if (done)
      {
        updateCommentFragment();
        stopProgress();
      }
    }
    else if (msgID == tagImageGetID)
    {
      if (success)
      {
        DbHandlerScaledImageRsp sir = (DbHandlerScaledImageRsp) response;

        if (Constants.USE_NEW_TAGVIEW_UI)
        {
          tagPhotoImgView.setTagImageBitmap(sir.bitmap);
          tagPhotoImgView.setHandleTouchEvent(true);
        }
        else
        {
          img_tagImage.setTagImageBitmap(sir.bitmap);
        }
      }
    }
  break;

  case DbHandlerConstants.DBMSG_DELETE:
    // once the tag is removed from the db, remove it
    // from the arraylist and update.
    comments.remove(removeCommentIndex);
    updateCommentFragment();
    stopProgress();
    Toast.makeText(TagViewActivity.this, "Comment Removed",
    Toast.LENGTH_SHORT).show();
  break;
  case DbHandlerConstants.DBMSG_ADD:
    if (response != null && response instanceof Comment)
    {
      Comment comm = (Comment) response;
    }
comments.add(comm);
position = comments.size() - 1;

if (position != null)
{
    updateCommentFragment();
    commentList.setSelection(position);
}
position = null;
commentTxt.setText("");
stopProgress();
String msg =
this.getResources().getString(R.string.toast_addcomment_success);
Toast.makeText(getApplicationContext(), msg,
Toast.LENGTH_SHORT).show();
img_commentImage.setImageDrawable(icon);
}
break;
}

/**
 * Implemented so that the sensors for the compass are only working when the application
 * is active and not just running in the background which would affect battery life
 * @see android.app.Activity#onResume()
 */
/**
 * Overrided onResume method. Currently, sensorManager code was causing crash.
 * DisplayTag must be called in this method as opposed to onCreate method to be assured that
 * fragments are created.
 * DisplayTag method populates fragments with information.
 */
@override
protected void onResume()
{

/**
 * Kale Commented this out because it was effecting launching this activity. Have to
 * investigate sensorManager.registerListener(this,
 * sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
 * SensorManager.SENSOR_DELAY_GAME); locationListener = new MyLocationListener();
 * locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
 * 0, locationListener); locationManager.requestLocationUpdates(LocationManager
 * .NETWORK_PROVIDER, 0, 0,
 * locationListener);
 * */
setupTag();
//displayTag(); // This must be called here so that onCreate has time to
//create fragments. Otherwise crash..
//retrieveComments(); //causing crash at getComments method
super.onResume();
}
* Implemented so that the sensors for the compass are only working when the application is active and not just running in the background which would affect battery life

* @see android.app.Activity#onPause()
/**
* SensorManager code to prevent crashes
*/
@Override
protected void onPause()
{
    // sensorManager.unregisterListener(this, sensorAccelerometer);
    // sensorManager.unregisterListener(this, sensorMagneticField);
    /*
    * Kale commented out because causing crashing
    *
    * sensorManager.unregisterListener(this);
    locationManager.removeUpdates(locationListener); locationListener = null;
    */
    super.onPause();
}
/*
* Method must be defined to implement SensorEventListener
*
* @see android.hardware.SensorEventListener#onAccuracyChanged(android.hardware.Sensor, int)
*/
@Override
public void onAccuracyChanged(Sensor arg0, int arg1)
{
    // TODO Auto-generated method stub
}
/*
* This method detects whenever the sensors pick up a change in location and/or position. When a change is detected, the compass will be updated to point in regards to the current location.
*
* Code modified from: http://sunil-android.blogspot.com/2013/02/create-our-android-compass.html
*
* @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent)
*/
/*
* @Override public void onSensorChanged(SensorEvent event) {
switch(event.sensor.getType()) //determine what sensor event has occurred
{ case Sensor.TYPE_MAGNETIC_FIELD: for(int i = 0; i < 3; i++) {
    valuesMagneticField[i] = event.values[i]; } break; case
Sensor.TYPE_ACCELEROMETER: for(int i = 0; i < 3; i++) {
    valuesAccelerometer[i] = event.values[i]; } break; }
* 
* boolean success = SensorManager.getRotationMatrix(matrixR, matrixI,
values.Accelerometer, values.MagneticField);

*  *
* if (success) { // if the rotation matrix was found above, update the compass {
  SensorManager.getOrientation(matrixR, matrixValues);
  // myCompass.update(matrixValues[0]);
  * compassTriangle.setRotation(matrixValues[0]);
  *}
*}

@Override
public void onSensorChanged(SensorEvent event)
{
  // spencerOnSensorChanged(event);
  // emilyOnSensorChanged(event);

  /**
   * Creates the context menu that allows the user to delete tags
   */
  @Override
  public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo)
  {
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;
    // show delete context menu only if user created the comment, or if the
    // comment is on their tag
    if (comments.get(info.position).getUsername().equalsIgnoreCase(UserSession.CURRENT_USER.getuName())
        || currentTag.getOwnerName().equalsIgnoreCase(UserSession.CURRENT_USER.getuName()))
    {
      menu.setHeaderTitle("Comment");
      menu.add(1, 1, 1, "Remove Comment");
    }
  }

  /**
   * Implements the click listeners for selecting an item from the context menu
   */
  @Override
  public boolean onContextItemSelected(android.view.MenuItem item)
  {
    final AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
    // delete the selected tag
    if (item.getItemId() == 1)
    {
      removeComment(info.position);
    }
    return true;
  }

  }
/*
 * FUNCTIONS
 */
private void removeComment(int index)
{
    removeCommentIndex = index;
    setupProgress(getString(R.string.progress_removing_comment));
    showProgress();
    mApp.sendMsgToDbHandler(rspHandler, this,
DbHandlerConstants.DBMSG_DELETE, comments.get(index));
}

Integer position = null; // Is this necessary? -SK 9/2

/**
 * Retrieves the comment for the tag by creating a new thread
 */
@Override
public void onCreateCommentViewCallback(boolean refresh)
{
    if (refresh || !commentsInitialized)
    {
        retrieveComments();
    }
    else
    {
        updateCommentFragment();
    }
}

private void retrieveComments()
{
    // this should be commentViewFragmentComments
    if (currentTag == null)
    {
        Log.e(TAG, "currentTag is NULL!");
        return;
    }

    setupProgress(getString(R.string.progress_retrieving_comments));
    showProgress();

    mApp.sendMsgToDbHandler(rspHandler, this,
DbHandlerConstants.DBMSG_GET_TAG_COMMENTS, currentTag);
}

/**
 * Displays the tag in new UI for viewing by the user
 */
@Override
public void onCreateDescriptionViewCallback()
{
    setupTag();
    displayTag();
*/
* Displays the tags information in the appropriate fragment
*/

@SuppressWarnings("unchecked")
// for unchecked cast from Serializable to ArrayList<Tag>
private void setupTag()
{
    Intent i = getIntent();
    if (i != null && i.hasExtra(EXTRA_TAGID))
    {
        currentTagID = i.getLongExtra(EXTRA_TAGID, 0);
        currentTag = null;
    }

    tagPhotoImgView = (ScaleImage) findViewById(R.id.tag_photo);
    tagPhotoImgView.setCallbacks(this);
    tagPhotoImgView.setHandleTouchEvents(false);

    /*
    * initialize Description View Fragment
    */
    descriptionTxtView = (TextView) ((DescriptionViewFragment)
            descriptionViewFragment).getView().findViewById(R.id.description);
    dateCreatedTxtView = (TextView) ((DescriptionViewFragment)
            descriptionViewFragment).getView().findViewById(R.id.label_date_created);
    timeStampTxtView = (TextView) ((DescriptionViewFragment)
            descriptionViewFragment).getView().findViewById(R.id.label_timestamp);
    lblLocationTxtView = (TextView) ((DescriptionViewFragment)
            descriptionViewFragment).getView().findViewById(R.id.label_location);
    locationTxtView = (TextView) ((DescriptionViewFragment)
            descriptionViewFragment).getView().findViewById(R.id.location);

    // Make a DB request for the Tag
    mApp.sendMsgToDbHandler(rsHandler, this,
            DbHandlerConstants.DBMSG_GET_TAG, new Tag(currentTagID));
}

private void displayTag()
{
    if (currentTag == null)
    {
        return;
    }

    // get location for MapViewHandler
    geo = currentTag.getLocation();

    // display tag name
    actionBar = getActionBar();
    actionBar.setTitle(currentTag.getName());

    // set Description in description fragment
    descriptionTxtView.setText(currentTag.getDescription());
}
Log.d(TAG, "Desc is " + currentTag.getDescription());

CacheDatabase tempcache = new CacheDatabase(this);
Tag temptag = tempcache.getMYTag(currentTag.getId());
Date date;
if (temptag != null)
  {  
    date = temptag.getCreatedDateTime();
  }
else
  {
    date = currentTag.getCreatedDateTime();
  }

SimpleDateFormat df = new SimpleDateFormat(\"DATE_FORMAT\");
SimpleDateFormat tf = new SimpleDateFormat(\"TIME_FORMAT\");
String fDate = df.format(date);
String fTime = tf.format(date);
// display date and time in description fragment
dateCreatedTxtView.setText("Date Created:");
timeStampTxtView.setText(fDate + " " + fTime);
// display location string in description fragment
lblLocationTxtView.setText("Location:");
locationTxtView.setText(lldf.format(geo.getLatitude()) + ", " +
lldf.format(geo.getLongitude()));

// display tag image
if (currentTag.getImageUrl() != null &
  & currentTag.getImageUrl().length() > 0)
  {
    String url = currentTag.getImageUrl();
    loadImage(url);
  }
Tag tag = null;
// Display Emotion
try
  {
    CacheDatabase cacheDatabase = new CacheDatabase(TagViewActivity.this);
tag = cacheDatabase.getMYTag(currentTag.getId());
int iirating = tag.getRatingScore();
if (iirating == 1)
  {
    imgve_smile.setBackgroundResource(R.drawable."");
  }
else if (iirating == 2)
imgve_smile.setBackgroundResource(R.drawable.smilee);

else
{
    imgve_smile.setBackgroundResource(R.drawable.smileg);
}
}

catch (Exception e)
{
    e.printStackTrace();
}

/**
 * Load the tag’s image from the URL and into the ImageView
 */
private void loadImage(String imgUrl)
{
    DbHandlerScaledImageReq gsi = new DbHandlerScaledImageReq();

    gsi.width = (int) (getResources().getDimension(R.dimen.image_width));
    gsi.height = (int) (getResources().getDimension(R.dimen.image_height));
    gsi.urls = new String[1];
    gsi.urls[0] = imgUrl;
    tagImageGetID = mApp.sendMsgToDbHandler(DbHandlerConstants.DBMSG_GET_SCALED_IMAGES, gsi);
}

/*
 * EVENT HANDLERS
 */
public class DeleteConfirmationDialogFragment extends DialogFragment
{
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState)
    {
        // Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setMessage(R.string.delete_tag_message).setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                Intent returnIntent = new Intent();
                returnIntent.putExtra(TAG_DELETED, currentTagID);
                setResult(RESULT_OK, returnIntent);
                finish();
            }
        }).setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
            }
        });

        // Create the AlertDialog object and return it

    }
return builder.create();
}
}

DeleteConfirmationDialogFragment confirmDelete = null;

static final int ACTIVITY_EDITTAG = 1;

/**
 * Handles the event of a user clicking on an item in the options menu
 */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
switch (item.getItemId())
{
  case R.id.action_edit:
    Intent intent = new Intent(this, AddTagActivity.class);
    intent.putExtra(Constants.EXTRA_TAGID, currentTag.getId());
    startActivityForResult(intent, ACTIVITY_EDITTAG);
    return true;
  case R.id.action_delete:
    FragmentManager manager = TagViewActivity.this.getFragmentManager();
    confirmDelete = new DeleteConfirmationDialogFragment();
    confirmDelete.show(manager, null);
    return true;
}
return super.onOptionsItemSelected(item);
}

@Override
public boolean doubleClickCallback()
{
String url = currentTag.getImageUrl();
if (url == null || url.length() == 0)
  return false;

int visibility;
visibility = lowerArea.getVisibility() == View.GONE ? View.VISIBLE : View.GONE;
lowerArea.setVisibility(visibility);

return true;
}

/**
 * Handle activity results here. Specifically when the edit tag activity returns. Will need to
 * update the Tag data.
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
if (requestCode == ACTIVITY_EDITTAG)
{
  if (resultCode == RESULT_OK)
    {
      if (requestCode == RESULT_OK)
    {

68
Add Tag XML

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    style="@style/GeotaggerLookAndFeel Activity"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight=".9"
        android:orientation="vertical">
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingLeft="@dimen/Scrollview_Leftmargin"
            android:paddingRight="@dimen/Scrollview_Leftmargin">
            <EditText
                android:id="@+id/addtag_name"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="left"
                android:layout_marginTop="@dimen/scrollview_topmargin"
                android:background="@drawable/select_text"
                android:ems="10"
                android:gravity="left"
                android:hint="@string/tag_name"
                android:inputType="textCapSentences"
                android:maxLength="35"
                android:maxLines="1"
                android:padding="@dimen/dataentry_textpadding"
                android:textSize="@dimen/dataentry_textsize">
                <requestFocus />
            </EditText>
        </LinearLayout>
    </ScrollView>
    <fragment
        android:id="@+id/addcomment_image_fragment"
        android:name="com.hci.geotagger.activities.AddImageFragment"
```
android:layout_width="fill_parent"
android:layout_height="@dimen/addtag_image_height"

<EditText
    android:id="@+id/addtag_desc"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_marginTop="@dimen/scrollview_topmargin"
    android:layout_weight="2.11"
    android:background="@drawable/select_text"
    android:ems="10"
    android:gravity="top"
    android:hint="@string/tag_description"
    android:inputType="textCapSentences|textMultiLine"
    android:lines="4"
    android:padding="@dimen/dataentry_textpadding"
    android:singleLine="false"
    android:textSize="@dimen/dataentry_textsize" />

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/scrollview_topmargin"
    android:gravity="left"
    android:orientation="horizontal">
    <TextView
        style="@style/GeotaggerLookAndFeelFieldLabelText"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:gravity="center_vertical"
        android:text="@string/location" />
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="left"
        android:orientation="vertical">
    </LinearLayout>
</LinearLayout>

<TextView
    style="@style/GeotaggerLookAndFeelFieldLabelText"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:gravity="center_vertical"
    android:text="@string/location" />

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
</LinearLayout>

<LinearLayout
    android:id="@+id/origlocationlayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <CheckBox
        android:id="@+id/origlocationcheckbox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/Scrollview_Leftmargin" />
</LinearLayout>

<CheckBox
    android:id="@+id/origlocationcheckbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="@dimen/Scrollview_Leftmargin" />

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="left"