

A
PROJECT REPORT
ON
POLARITY ANALYSER USING NLP

Submitted in partial fulfillment of the requirements for the award of the degree

Of
BACHELOR OF TECHNOLOGY

In
COMPUTER SCIENCE AND ENGINEERING

BY
G U BHARATH CHANDRA , NITHIN REDDY, GOMPAVAMSIKRISHNA
RA1411003040060 , RA1411003040022 , RA1411003040100

Under the Guidance of

Mr C. Sabarinathan, B. Tech, M.E, Asst. Professor

Department of Computer Science and Engineering

CS1049 MINOR PROJECT

SEMESTER VI – April 2017



FACULTY OF ENGINEERING AND TECHNOLOGY
SRM UNIVERSITY

Vadapalani Campus, Chennai -26

SRM UNIVERSITY

(UNDER SECTION 3 OF THE UGC ACT, 1956)

VADAPALANI CAMPUS

CHENNAI-600 026

BONAFIDE CERTIFICATE

REGISTER NO _____

Certified to be the Bonafide Record of work done by
_____ of _____ year **B.TECH** degree
course in the practical CS1049-MINOR PROJECT in **SRM UNIVERSITY**,
Vadapalani Campus, Chennai - 26 during the academic year 2016-2017.

Date:

Faculty Incharge

Head of the Department

Submitted for University Examination held in
_____ **SRM UNIVERSITY at Vadapalani.**

External Examiner 1

External Examiner 2

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|--------------------|------------------------------|-----------------|
| | ACKNOWLEDGEMENT | 3 |
| | ABSTRACT | 4 |
| | LIST OF FIGURES | 5 |
| | LIST OF ABBREVIATIONS | 5 |
| 1. | INTRODUCTION | |
| | 1.1 General | 6 |
| | 1.2 Objective | 7 |
| | 1.3 Description | 8 |
| 2. | LITERATURE REVIEW | 9 |
| 3. | SYSTEM ANALYSIS | |
| | 3.1 Existing System | 13 |
| | 3.2 Proposed System | 18 |
| 4. | PROJECT DESCRIPTION | |
| 4.1 | Problem Definition | 19 |
| 4.2 | Methodology | 20 |
| 4.3 | Operation | 21 |
| 4.4 | System Analysis | 22 |
| 5. | SYSTEM REQUIREMENTS | |
| 5.1 | Hardware Requirements | 24 |
| 5.2 | Software Requirements | 25 |
| 5.2.1 | Front End Description | 26 |
| 5.2.2 | Back End Description | 28 |
| 5.3 | Specific Requirements | 29 |
| 6. | SYSTEM DESIGN | |
| 6.1 | System Architecture Diagram | 32 |
| 6.2 | Class Diagram | 33 |
| 6.3 | UML Diagram | 34 |
| 6.4 | State Diagram | 35 |

| | | |
|-----|---------------------------|-----------|
| 7 | TESTING | |
| 7.1 | Screenshots | 36 |
| 7.2 | Data Structures | 39 |
| 8. | APPLICATION | |
| 8.1 | General | 40 |
| 8.2 | Specific Applications | 41 |
| 9. | FUTURE ENHANCEMENT | 44 |
| 10 | CONCLUSION | 45 |
| | SOURCE CODE | 46 |
| | REFERENCES | 63 |

ACKNOWLEDGEMENT

We place our deep sense of gratitude to our beloved **Chancellor Mr .T. R. PACHAMUTHU, SRM UNIVERSITY** for providing us with the requisite infrastructure throughout the course.

We take the opportunity to extend our heartfelt thanks to our respected **DEAN, Dr . K. DURAIVELU** for his support and impeccable guidance.

We are extremely grateful to the **Head of the Department MRS.PADMAVATHI** for having encouraged and helped us throughout the course of our project. Without her supervision and feedback ,it would have been really hard for us to finish our project in a timely manner. Thus, we feel deeply obliged for the support.

We are also grateful to our **GUIDE, MR . C SABARINATHAN** for having assisted and mentored us so diligently in the process of preparing our project. Without his persistent support and co-operation , we couldn't have accomplished our ideas.

Abstract

This approach regulates the polarity of appraisal which belongs to distinct domains by availing scrutinizer algorithm where the stop words like[if,the,it,....etc] are breaked apart by using natural language processing library, where ,the leftover words are contemplated as independent tokens now they endure a process called tokenization [which actually represents the tangeable part of word] these are now parsed along the java wiktionary library which is a multilingual dictionary by which the authentic meaning of the word is determined which can eventually determine the polarity of the content by employing count vectorization mechanism a bipartite graph is cancor ted for the pre-resulted words which can ultimately proffer the contradiction of a review where the graph is actually organized between the digit of the word in a sentence at x-axis and the affirmative and the cynical barriers at y-axis where the potency will be plotted with scale of +0.5 for positive and -0.5 for negative words which regulates the graph scale and polarity of the sentence.

LIST OF FIGURES

| FIGURE NO. | NAME OF THE FIGURE | PAGE NO. |
|------------|-----------------------------|----------|
| 6.1 | System Architecture Diagram | 32 |
| 6.2 | Class Diagram | 33 |
| 6.3 | UML Diagram | 34 |
| 6.4 | State Diagram | 35 |
| 7.1.1 | Home page 1 | 36 |
| 7.1.2 | Home page 2 | 36 |
| 7.1.3 | analysis page | 37 |
| 7.1.4 | User details page | 37 |
| 7.1.5 | graph analysis page | 38 |

1. Introduction

1.1 General

In this paper, we propose learning sentiment-specific word embeddings to confer sentiment embeddings for sentiment analysis. We retain the effectiveness of word contexts and accomplish sentiment of texts for learning more powerful continuous word representations. By capturing both context and sentiment level evidences, the nearest neighbors in the embedding space are not only semantically similar but also favor to have the same sentiment polarity, so that it is able to separate good and bad to opposite ends of the spectrum. In order to learn sentiment embeddings effectively, we develop a number of prototypes to capture sentiment of texts (e.g. sentences and words) as well as contexts of words with dedicated loss functions. We learn sentiment embeddings from tweets, leveraging positive and negative emoticons as pseudo sentiment labels of sentences without manual annotations. We obtain lexical level sentiment supervision from Urban Dictionary based on a small list of sentiment seeds with minor manual annotation. In existing system there are only 2 mechanisms as NLP and pos-tagging where The polarity of reviews which belongs to specific domains are processed with NLP and the sentiment of the sentence is determined by sentiment analysis and both the context and sentiment of the sentence by sentiment word embeddings. but this system fails in conditions where the context of the word is different but the sentiment is same since the system neighbours them together which results in loss of precision.

1.2 Objective

The objective of this system is to determine domain independent and dependent words using a naive bayes type classifier and increase the precision by using scrutinizer algorithm where the stop words like (if , the , it ..Etc..) are seperated by using natural language processing library. The rest of the words are considered as independent tokens. Now they go through a process called tokenization. part-of-speech tagging(POS tagging or POST), also called grammatical tagging or word category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context i.e.,its relationship with adjacent and related words in a phrase, sentence, or

paragraph and move on projecting the details in the form of a bar graph with the provided pre-result as an input.

2. Literature review

Existing embedding learning approaches are mostly on the basis of distributional hypothesis, which states that the representations of words are reflected by their contexts. As a result, words with similar grammatical usages and semantic meanings, such as “hotel” and “motel”, are mapped into neighboring vectors in the embedding space.

Since word embeddings capture semantic similarities between words, they have been leveraged as inputs or extra word features for a variety of natural language processing tasks

Learning Sentiment-specific word embedding for twitter sentiment classification

We present a method that learns word embedding for Twitter sentiment classification in this paper. Most existing algorithms for learning continuous word representations typically only model the syntactic context of words but ignore the sentiment of text. This is problematic for sentiment analysis as they usually map words with similar syntactic context but opposite sentiment polarity, such as good and bad, to neighboring word vectors. We address this issue by learning sentiment-specific word embedding (SSWE), which encodes sentiment information in the continuous representation of words. Specifically, we develop three neural networks to effectively incorporate the supervision from sentiment polarity of text (e.g. sentences or tweets) in their loss functions. To obtain large scale training

corpora, we learn the sentiment-specific word embedding from massive distant-supervised tweets collected by positive and negative emoticons. Experiments on applying SSWE to a benchmark Twitter sentiment classification dataset in SemEval 2013 show that (1) the SSWE feature performs comparably with hand-crafted features in the top-performed system; (2) the performance is further improved by concatenating SSWE with existing feature set.

Cooooolll: A deep learning system for twitter sentiment classification

In this paper, we develop a deep learning system for message-level Twitter sentiment classification. Among the 45 submitted systems including the SemEval 2013 participants, our system (Cooooolll) is ranked 2nd on the Twitter2014 test set of SemEval 2014 Task 9. Cooooolll is built in a supervised learning framework by concatenating the sentiment-specific word embedding (SSWE) features with the state-of-the-art hand-crafted features. We develop a neural network with hybrid loss function 1 to learn SSWE, which encodes the sentiment information of tweets in the continuous representation of words. To obtain large-scale training corpora, we train SSWE from 10M tweets collected by positive and negative emoticons, without any manual annotation. Our system can be easily re-implemented with the publicly available sentiment-specific word embedding.

A neural probabilistic language model

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the **curse of dimensionality**: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Traditional but very successful approaches based on n-grams obtain generalization by concatenating very short overlapping sequences seen in the training set. We propose to fight the curse of dimensionality by **learning a distributed representation for words** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously (1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence. Training such large models (with millions of parameters) within a reasonable time is itself a significant challenge. We report on experiments using neural networks for the probability function, showing on two text corpora that the proposed approach significantly improves on state-of-the-art n-gram models, and that the proposed approach allows to take advantage of longer contexts.

Distributed representations of words and phrases and their compositionality

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number

of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

An inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases. For example, the meanings of "Canada" and "Air" cannot be easily combined to obtain "Air Canada". Motivated by this example, we present a simple method for finding phrases in text, and show that learning good vector representations for millions of phrases is possible.

Word alignment modeling with context dependent deep neural network

In this paper, we explore a novel bilingual word alignment approach based on DNN (Deep Neural Network), which has been proven to be very effective in various machine learning tasks (Collobert et al., 2011). We describe in detail how we adapt and extend the CD-DNNHMM (Dahl et al., 2012) method introduced in speech recognition to the HMMbased word alignment model, in which bilingual word embedding is discriminatively learnt to capture lexical translation information, and surrounding words are leveraged to model context information in bilingual sentences. While being capable to model the rich bilingual correspondence, our method generates a very compact model with much fewer parameters. Experiments on a large scale English- Chinese word alignment task show that the proposed method outperforms the HMM and IBM model 4 baselines by 2 points in F-score.

3. System analysis

3.1 EXISTING SYSTEM:

- Existing embedding learning approaches are mostly on the basis of distributional hypothesis, which states that the representations of words are reflected by their contexts. As a result, words with similar grammatical usages and semantic meanings, such as “hotel” and “motel”, are mapped into neighboring vectors in the embedding space.
- Since word embeddings capture semantic similarities between words, they have been leveraged as inputs or extra word features for a variety of natural language processing tasks.
- Mnih and Hinton introduce a log-bilinear language model.
- Collobert and Weston train word embeddings with a ranking-type hinge loss function by replacing the middle word within a window with a randomly selected one.
- Mikolov et al. introduce continuous bag-of-words (CBOW) and continuous skip-gram, and release the popular word2vec3 toolkit. CBOW model predicts the current word based on the embeddings of its context words, and Skip-gram model predicts surrounding words given the embeddings of current word.
- Mnih and Kavukcuoglu accelerate the embedding learning procedure with noise contrastive estimation.

DISADVANTAGES OF EXISTING SYSTEM:

- The most serious problem of context-based embedding learning algorithms is that they only model the contexts of words but ignore the sentiment information of text. As a result, words with opposite polarity, such as good and bad, are mapped into close vectors in the embedding space.
- Existing word embedding learning algorithms typically only use the contexts of words but ignore the sentiment of texts.

3.2 PROPOSED SYSTEM:

- In this paper, we propose learning sentiment-specific word embeddings dubbed sentiment embeddings for sentiment analysis. We retain the effectiveness of word contexts and exploit sentiment of texts for learning more powerful continuous word representations.
- By capturing both context and sentiment level evidences, the nearest neighbors in the embedding space are not only semantically similar but also favor to have the same sentiment polarity, so that it is able to separate good and bad to opposite ends of the spectrum.
- We learn sentiment embeddings from tweets, leveraging positive and negative emoticons as pseudo sentiment labels of sentences without manual annotations. We obtain lexical level sentiment supervision from Urban Dictionary based on a small list of sentiment seeds with minor manual annotation.
- We propose learning sentiment embeddings that encode sentiment of texts in continuous word representation.

- We learn sentiment embeddings from tweets with positive and negative emoticons as distant-supervised corpora without any manual annotations.
- We verify the effectiveness of sentiment embeddings by applying them to three sentiment analysis tasks. Empirical experimental results show that sentiment embeddings outperform context-based embeddings on several benchmark datasets of these tasks.

ADVANTAGES OF PROPOSED SYSTEM:

We evaluate the effectiveness of sentiment embeddings empirically by applying them to three sentiment analysis tasks.

Word level sentiment analysis on benchmark sentiment lexicons can help us see whether sentiment embeddings are useful to discover similarities between sentiment words.

Sentence level sentiment classification on tweets and reviews help us understand whether sentiment embeddings are helpful in capturing discriminative features for predict the sentiment of text.

Building sentiment lexicon is useful for measuring the extent to which sentiment embeddings improve lexical level tasks that need to find similarities between words.

Experimental results show that sentiment embeddings consistently outperform context-based word embeddings, and yields state-of-the-art performances on several benchmark datasets of these tasks.

4. Project description

4.1 Problem definition

We present the methods for learning POLARITYANALYSER in this section. We first describe standard context-based methods for learning word embeddings. Afterwards, we introduce our extension for capturing sentiment polarity of sentences before presenting hybrid models which encode both sentiment and context level information. We then describe the integration of word level information for embedding learning. We present the approaches to encode sentiment polarity of sentences in sentiment embeddings in this part. We describe two methods including a prediction model and a ranking model to take considerations of sentiment of sentences.

4.3 OPERATION

Back-end:

- Generally the relational database uses the tables to represent the information which it handles. A table consists of rows and columns; row is identified with record whereas column is identified with field name, so each column holds a single value of a predefined data type. Generally these data types can be text data, numeric data, dates, and binary data such as images and sound as per our requirement. A dedicated language called Structured Query Language is used to access the data.
- Java Server Pages has Standard Tag Library which includes the number of actions for the database access to improve the simple database-driven Java Server Page applications. Basically these actions are used to provide the following features:

1. Using a connection pool for better performance and scalability.
2. The features are to support the queries, updates, and insertion process.
3. To handle the most common data-type conversions.
4. To Support a combination of databases.

4.3

METHODOLOGY

a] PREDICTION MODEL

The basic idea of the prediction model is regarding sentiment prediction as a multi-class classification task. It predicts positive/negative categorical probabilities of a word sequence by regarding word embeddings as parameters. In particular, given a variable sized sentence, we slide fixed window of words across a sentence and predict the sentiment polarity of each window based on local word embeddings. This assumption is suitable for tweets if window size is large, since tweets are typically short and sentiment condensed. However, it might be not appropriate for document level review texts where sentiment shifting indicators (e.g., negation and contrast) are frequently used.

b] RANKING MODEL

We describe an alternative of prediction model, which is a ranking model that outputs two real-valued sentiment scores for a word sequence with fixed window size. The basic idea of ranking model is that if the gold sentiment polarity of a word sequence is positive, the predicted positive score should be higher than the negative score. Similarly, if the gold sentiment polarity of a word sequence is negative, its positive score should be smaller than the negative score. For example, if a word sequence is associated with two scores [frank(pos), frank(neg)], then the values of [0.7, 0.1] can be interpreted as a positive case because the positive score 0.7 is greater than the negative score 0.1. By that analogy, the result with [0.2, 0.6] indicates a negative polarity.

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?

-The dialog to guide the operating personnel in providing input. Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the systems relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

MODULES

OSN Entity:

In this module, first we develop the OSN entity. Here we introduce two entities called Admin and User. The users view the Uploaded product details and give reviews to that products and users expose their emotions by the way of emojis. The Emojis express exactly what the user thought about the product. Its very ease to understand its help other users to decide. They have also view their own product search history. They view also all products uploaded by the Admin. Admin upload the products under the products domain for the users. Admin can view all uploaded products .Here admin is view domains the sentiment analysis of the Users reviews and emojis. And also admin can view the users search history it helps the admin to know about their users wish list. Admin view the negative and positive review of the particular domain products. And also the admin generate a graph analyzation of the negative and positive reviews of the domain products. It helps to understand the domains status.

Classifier :

The base module of the project , it classifies the information on required basis, where it determines wether the data is domain specific or not hence, if the word is not domain specific then it may as well be irrelevant word or type mismatched word so it may result in false polarity of the data, hence the classifier dismisses the word making the system invulnerable, and reduces the time complexity anyway. but if the word is domain specific the it is considered in the sentence and carried through the further processes.

Count vectorization mechanism:

Is a process of rewriting a loop where every word undergoes the process of tokenization where every word is treated as a token independently and parsed within the java Wiktionary library to obtain the polarity of the words where each word is processed by determining the synonymns and ultimately driving the words towards any specific polarity.

d] Bipartite graph :

It is plotted for the preresulted words which can ultimately provide the polarity of the review where the graph is actually plotted between the number of the words in sentence at X-axis and the positive and negative barriers at Y-axis , where the intensity will be plotted with scale of +0.5 for positive words and -0.5 for negative words which determines the graph scale and ultimately determine the polarity of the sentence.

e] Sentiment Lexicon:

Sentiment embeddings to building sentiment lexicon, which is useful for measuring the extent to which sentiment embeddings improve lexical level tasks that need to find similarities between words. sentiment lexicon learning as a word-level classification task, which consists of two part:

(1) An embedding learning algorithm to effectively learn the continuous representation of words, which are used as features for word-level sentiment classification,

(2) A seed expansion algorithm that expands a small list of sentiment seeds to collect training data for building the word-level classifier. The learned sentiment embeddings are naturally regarded as continuous word features.discovering similarities between sentiment words. On sentence level sentiment classification, sentiment embeddings are helpful in capturing discriminative features for predicting the sentiment of sentences. On lexical level task like building sentiment lexicon, sentiment embeddings are shown to be

useful for measuring the similarities between words. Hybrid models that capture both context and sentiment information are the best performers on all three tasks.

SYSTEM ANALYSIS

EXISTING SYSTEM:

Existing embedding learning approaches are mostly on the basis of distributional hypothesis, which states that the representations of words are reflected by their contexts. As a result, words with similar grammatical usages and semantic meanings, such as “hotel” and “motel”, are mapped into neighboring vectors in the embedding space. Since word embeddings capture semantic similarities between words, they have been leveraged as inputs or extra word features for a variety of natural language processing tasks. Mnih and Hinton introduce a log-bilinear language model. Collobert and Weston train word embeddings with a ranking-type hinge loss function by replacing the middle word within a window with a randomly selected one. Mikolov et al. introduce continuous bag-of-words (CBOW) and continuous skip-gram, and release the popular word2vec3 toolkit. CBOW model predicts the current word based on the embeddings of its context words, and Skip-gram model predicts surrounding words given the embeddings of current word. Mnih and Kavukcuoglu accelerate the embedding learning procedure with noise contrastive estimation.

DISADVANTAGES OF EXISTING SYSTEM

The most serious problem of context-based embedding learning algorithms is that they only model the contexts of words but ignore the sentiment information of text. As a result, words with opposite polarity, such as good and bad, are mapped into close vectors in the embedding space. Existing word embedding learning algorithms typically only use the contexts of words but ignore the sentiment of texts.

PROPOSED SYSTEM

In this paper, we propose learning sentiment-specific word embeddings dubbed sentiment embeddings for sentiment analysis. We retain the effectiveness of word contexts and exploit sentiment of texts for learning more powerful continuous word representations. By capturing both context and sentiment level evidences, the nearest neighbors in the embedding space are not only semantically similar but also favor to have the same sentiment polarity, so that it is able to separate good and bad to opposite ends of the spectrum. We learn sentiment embeddings from tweets, leveraging positive and negative emoticons as pseudo sentiment labels of sentences without manual annotations. We obtain lexical level sentiment supervision from Urban Dictionary based on a small list of sentiment seeds with minor manual annotation. We propose learning sentiment embeddings that encode sentiment of texts in continuous word representation. We learn sentiment embeddings from tweets with positive and negative emoticons as distant-supervised corpora without any manual annotations. We verify the effectiveness of sentiment embeddings by applying them to three sentiment analysis tasks. Empirical experimental results show that sentiment embeddings outperform context-based embeddings on several benchmark datasets of these tasks.

ADVANTAGES OF PROPOSED SYSTEM

We evaluate the effectiveness of sentiment embeddings empirically by applying them to three sentiment analysis tasks. Word level sentiment analysis on benchmark sentiment lexicons can help us see whether sentiment embeddings are useful to discover similarities between sentiment words. Sentence level sentiment classification on tweets and reviews help us understand whether sentiment embeddings are helpful in capturing discriminative features for predict the sentiment of text. Building sentiment lexicon is useful for measuring the extent to which sentiment embeddings improve lexical level tasks that need to find similarities between words. Experimental results show that sentiment embeddings

consistently outperform context-based word embeddings, and yields state-of-the-art performances on several benchmark datasets of these tasks.

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

IMPLEMENTATION

OPEN NLP

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text.

It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. These tasks are usually required to build more advanced text processing services. OpenNLP also includes maximum entropy and perceptron based machine learning.

TOKENIZATION

The OpenNLP Tokenizers segment an input character sequence into tokens. Tokens are usually words, punctuation, numbers, etc.

OpenNLP offers multiple tokenizer implementations:

Whitespace Tokenizer - A whitespace tokenizer, non white space sequences are identified as tokens

Simple Tokenizer - A character class tokenizer, sequences of the same character class are tokens

Learnable Tokenizer - A maximum entropy tokenizer, detects token boundaries based on probability model

POS TAGGING

Part-of-speech tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech such as noun, verb, adjective, etc., based on its definition, as well as its context – i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph.

THE MODEL FILES ARE:

POS tagger – en-pos-maxent.bin

Sentence Detector – en-sent.bin

JFREE CHART

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

Map Visualization

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world.

The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

JDBC Goals

JDBC drove the development of the API. The goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The design goals for JDBC are as follows:

SQL Level API

We felt that the main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

1. JDBC must be implemented on top of common database interfaces

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

2. Provide a Java interface that is consistent with the rest of the Java system

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

3. Using strong, static typing wherever possible

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

4. Keeping the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be

simple to perform with JDBC. However, more complex SQL statements should also be possible.

NAIVE BAYES CLASSIFIER

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying **Bayes theorem** with strong (naive) independence assumptions between the features. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible **correlations** between the color, roundness, and diameter features.

Inspired from the above algorithm we implemented a NB classifier which is the base module of the project , it classifies the information on required basis, where it determines whether the data is domain specific or not if is not domain specific then it may as well be irrelevant word or type mismatched which result in false polarity of the data.

Java Wiktionary library

JWKTL is an application programming interface for the free multilingual online dictionary Wiktionary. Wiktionary is collaboratively constructed by volunteers and continually growing. JWKTL enables efficient and structured access to the information encoded in the English, the German, and the Russian Wiktionary language editions, including sense definitions, part of speech tags, etymology, example sentences, translations, semantic relations, and many other lexical information types. The API was first described in an LREC 2008 paper. The Russian JWKTL parser is based on Wikokit. Each token is parsed through this jwl for the sentiment of the specific word.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Core i3 2.4 GHz.
- Hard Disk : 40 GB.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

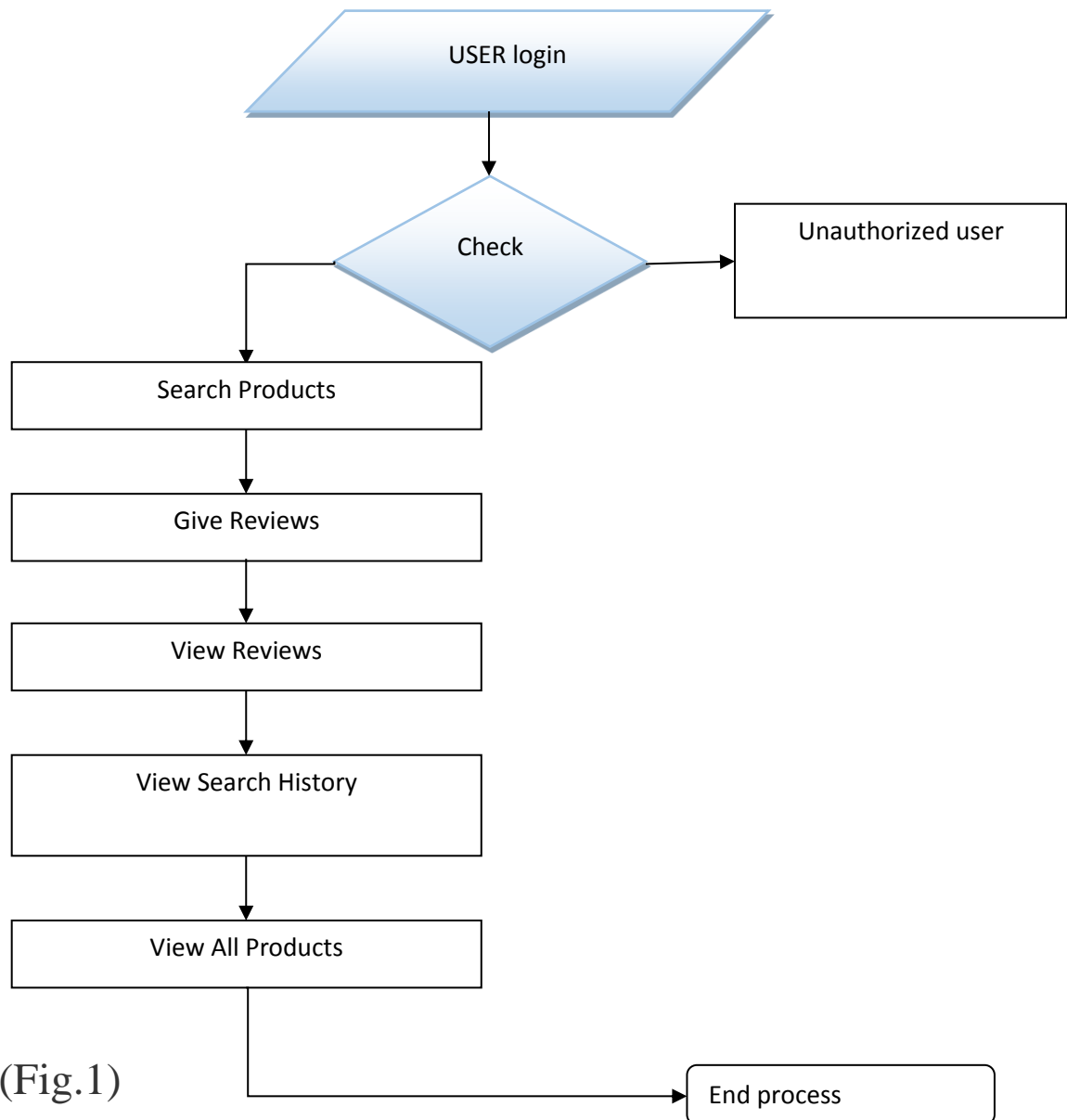
SOFTWARE REQUIREMENTS:

- Operating system : - Windows XP/7.
- Coding Language : JAVA/J2EE
- Data Base : MYSQL

6. System design

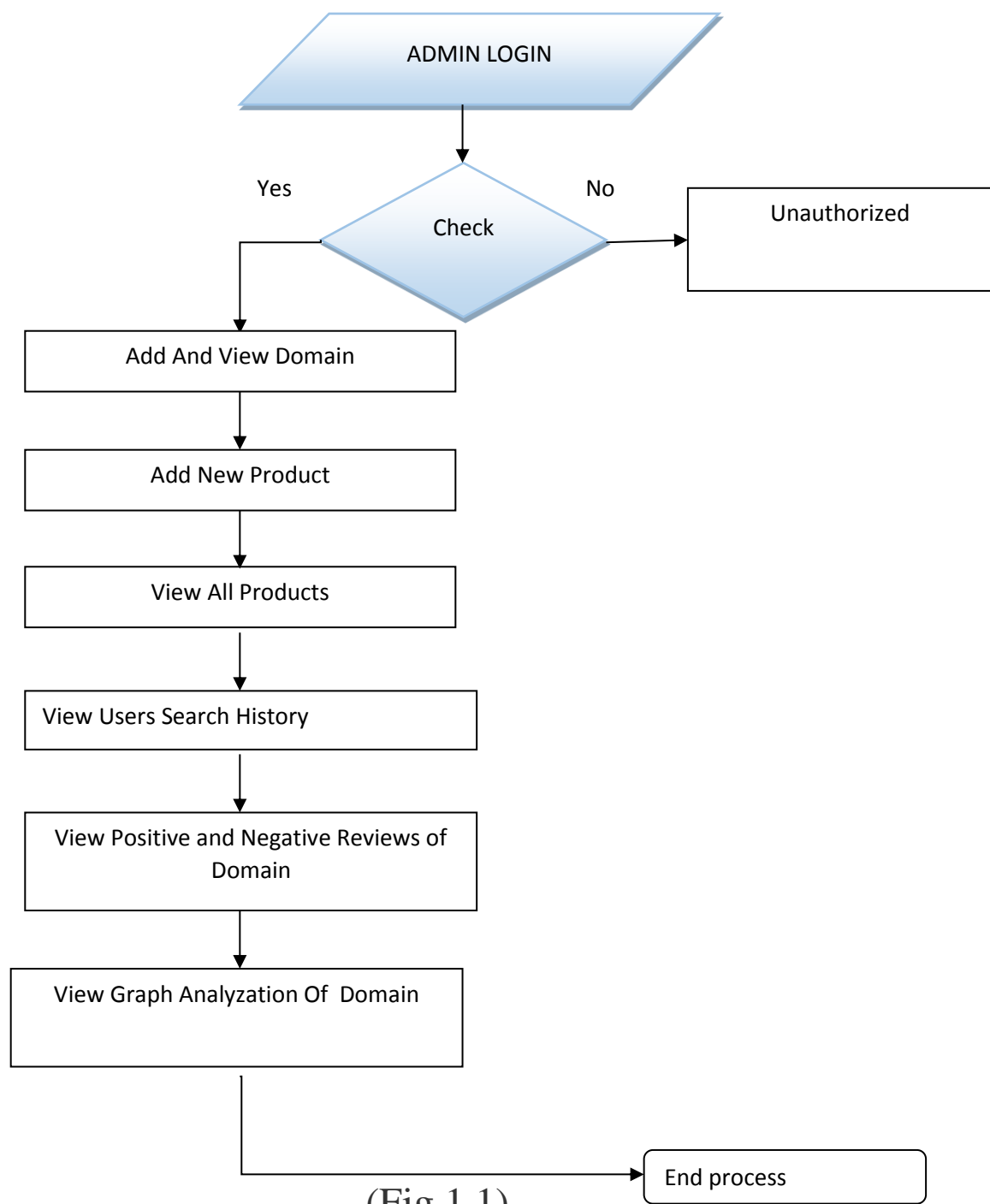
DATA FLOW DIAGRAM

a)

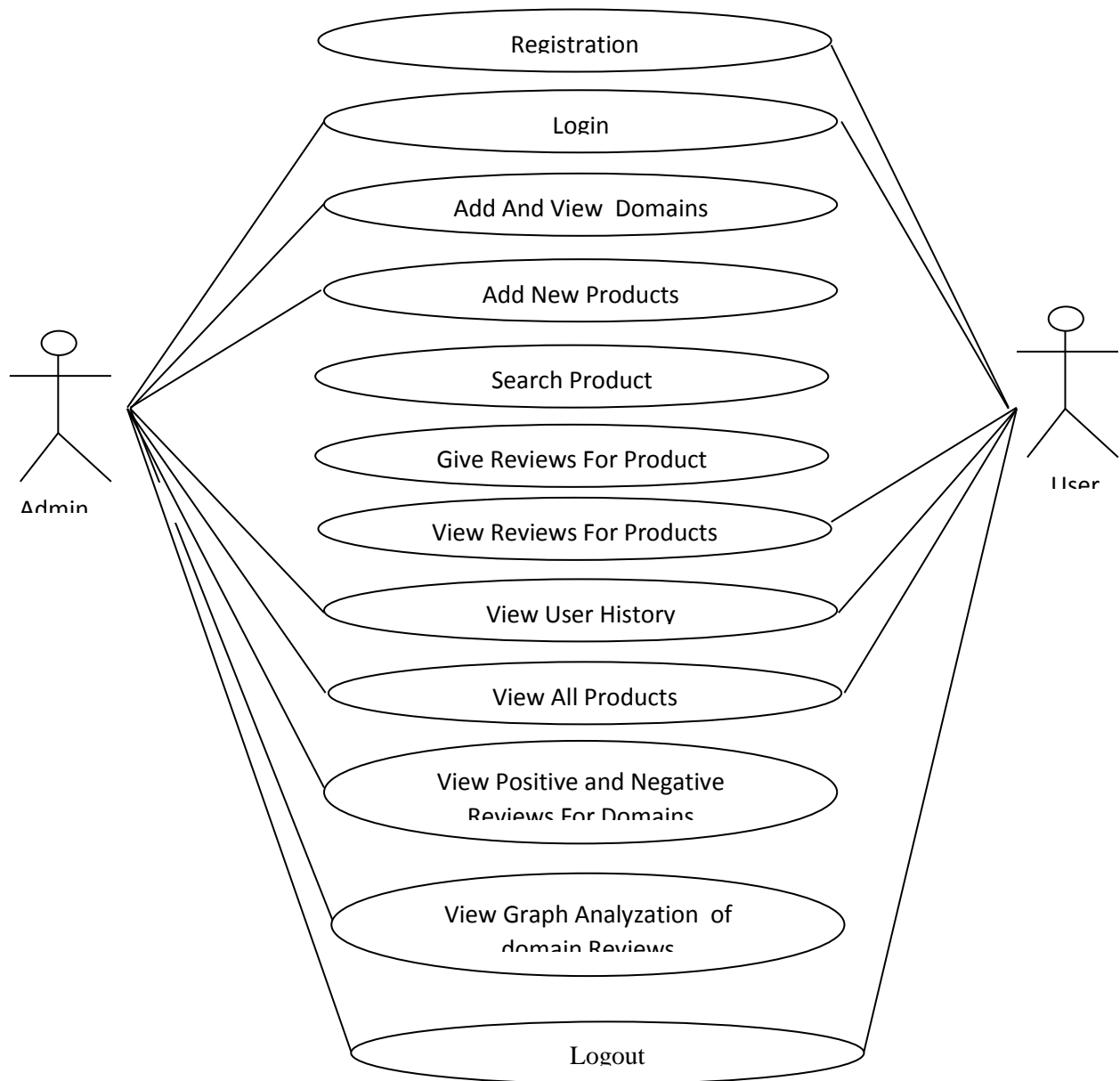


(Fig.1)

b]

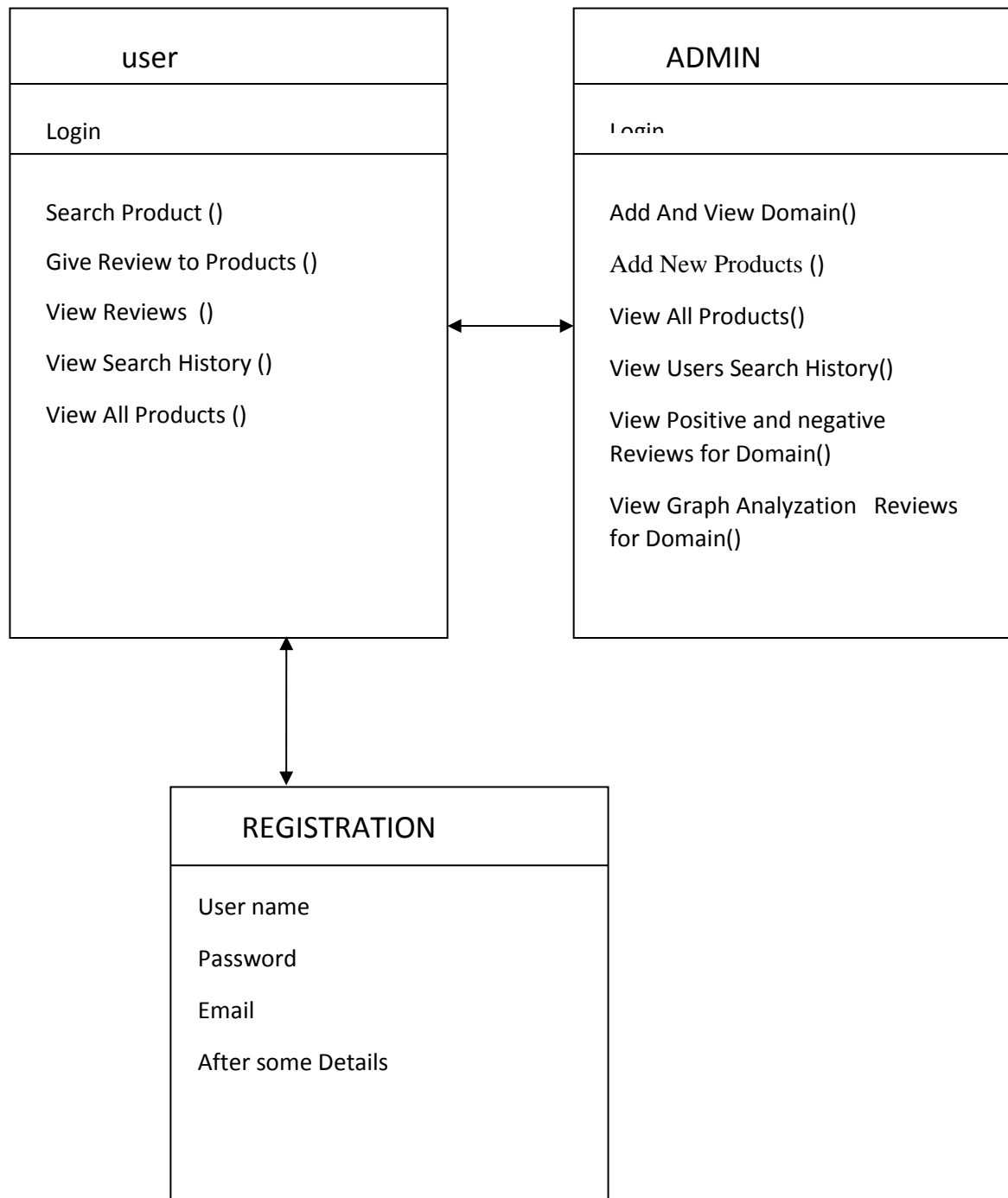


USE CASE DIAGRAM:

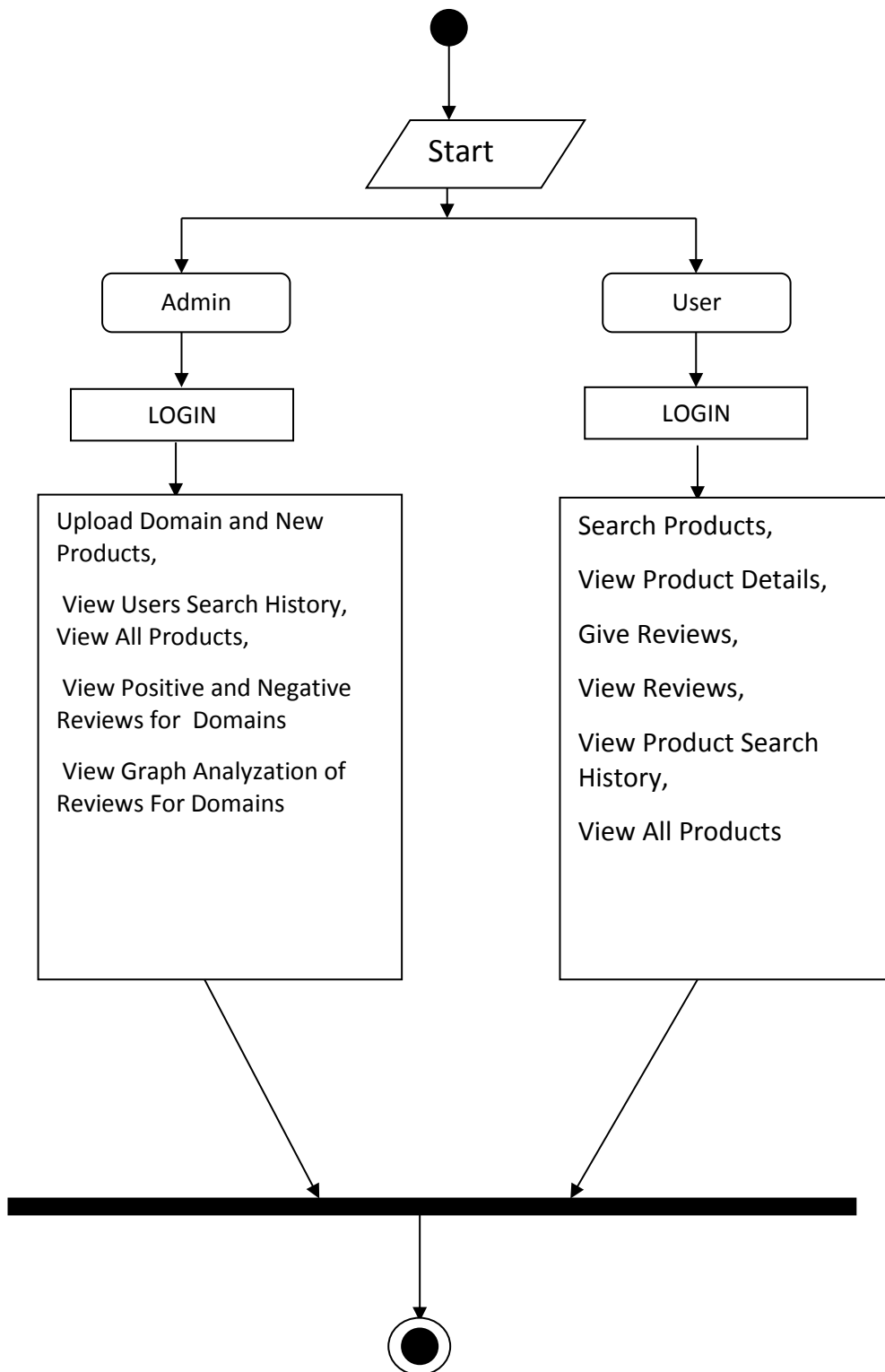


(Fig.2)

CLASS DIAGRAM

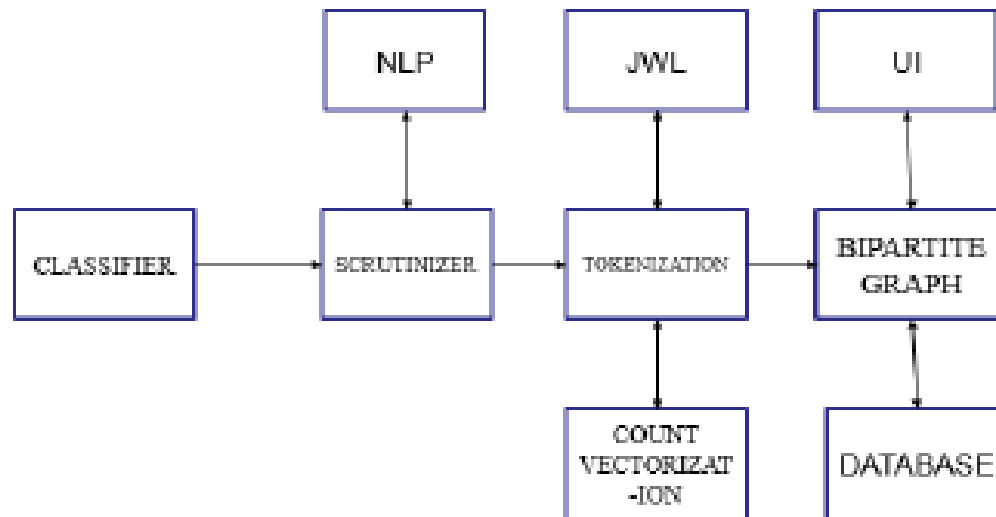


ACTIVITY DIAGRAM:



(Fig.5)

SYSTEM ARCHITECTURE:

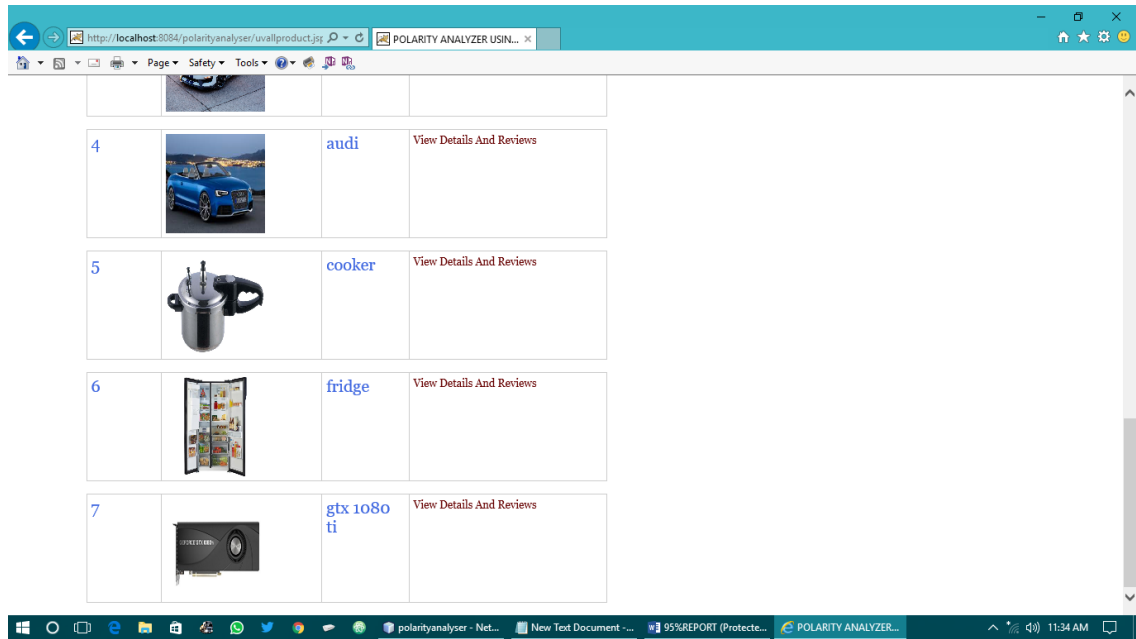


(Fig.6)

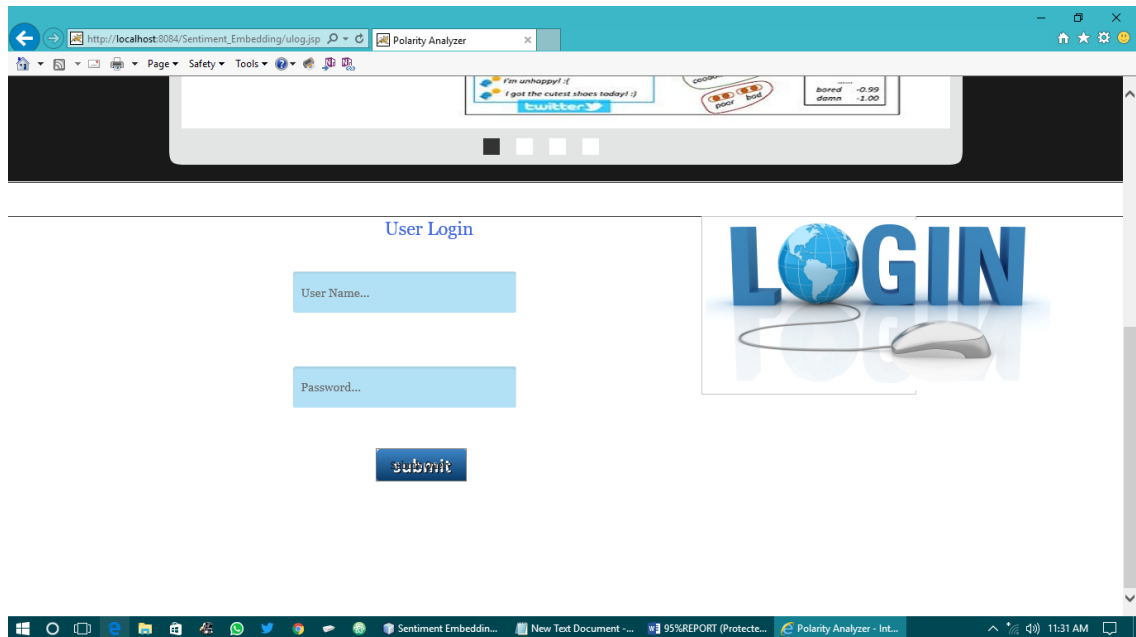
screenshot 1



2



3



4



Add New Domain

Enter New Domain Name:

submit

Side Menu Bar

- Home
- Add New Domain
- Add New Product
- View All Product
- View Users Search
- Users



Positive And Negative Reviews

Side Menu Bar

[Back](#)



CARS Positive Reviews

| | | |
|---|---------|-------------------|
| 3 | ferrari | super car.. |
| 4 | audi | good milage |
| 4 | audi | amazing speed.. |
| 4 | audi | beautiful stylish |

CARS Negative Reviews

| | | |
|---|------|--------------------|
| 4 | audi | damage seats works |
|---|------|--------------------|

Emoji Reviews

| | | |
|---|-----------|-------------------------------------------------------------------------------------|
| 3 | ferrari-> |  |
| 4 | audi-> |  |

6.

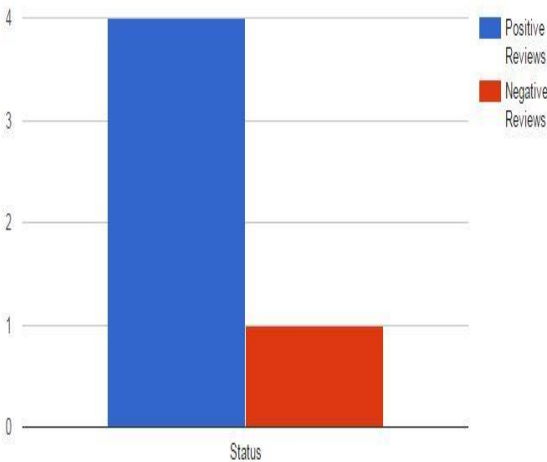


Graph Analyzation

Side Menu Bar

Back

CARS



Applications

1. Computing customer satisfaction metrics

You can get an idea of how happy customers are with your products from the ratio of positive to negative tweets about them.

2. Identifying detractors and promoters

It can be used for customer service, by spotting dissatisfaction or problems with products.

i) To forecast market movement based on news, blogs and social media sentiment

ii) To identify the clients with negative sentiment in social media or news and to increase the margin for transactions with them for default protection

Further enhancements:

There is a growing interest in deep sentiment analysis of text, in different areas of application. It is not enough to say that a text is overall positive or overall negative. Users would like to know which separate topics are talked about in the text, which of them are positive and which are negative. So I suppose there will be a trend towards greater use of NLP techniques (such as syntactic parsing, coreference resolution, etc), in addition to machine learning methods. At Revealed Context (the technology arm of Converseon), we also have intensity and confidence scores, as well as emotion and more. These new classifiers really dimensionalize the nuances of human expression in meaningful ways. There is also a move away from document/record level analysis of the text towards entity/facet level - meaning every expression of opinion is captured so that we can really understand the root cause drivers of opinions. This requires machine learning approaches that are superseding more traditional rules based approaches. In order to correctly understand the collective sentiment, we need to change the analysis paradigm (I personally believe is wrong). Instead of calculating and attribute a numerical value to sentiments, we will be better off if we assigned a spectrum of values to sentiments. The overall flavor of the documents will change from; Positive, Neutral or Negative, to a more comprehensive and colorful (human like) sentiment output. The tools and methods for this typology of challenges already exist and can be found in the shape of fuzzy logic.

10. CONCLUSION

We learn sentiment-specific word embeddings (named as sentiment embeddings) in this paper. Different from majority of exiting studies that only encode word contexts in word embeddings, we factor in sentiment of texts to facilitate the ability of word embeddings in capturing word similarities in terms of sentiment semantics. As a result, the words with similar contexts but opposite sentiment polarity labels like “good” and “bad” can be separated in the sentiment embedding space. We introduce several neural networks to effectively encode context and sentiment level informations simultaneously into word embeddings in a unified way. The effectiveness of sentiment embeddings are verified empirically on three sentiment analysis tasks. On word level sentiment analysis, we show that sentiment embeddings are useful for discovering similarities between sentiment words. On sentence level sentiment classification, sentiment embeddings are helpful in capturing discriminative features for predicting the sentiment of sentences. On lexical level task like building sentiment lexicon, sentiment embeddings are shown to be useful for measuring the similarities between words. Hybrid models that capture both context and sentiment information are the best performers on all three tasks.

SOURCE CODE

Main.jsp

```
<% @page import="network.DbConnection"%>
<% @page import="java.sql.ResultSet"%>
<% @page import="java.sql.Statement"%>
<% @page import="java.sql.Connection"%>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Sentiment Embeddings </title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
/>
<link rel="stylesheet" href="layout/styles/layout.css" type="text/css" />
<script type="text/javascript" src="layout/scripts/jquery.min.js"></script>
<script type="text/javascript"
src="layout/scripts/jquery.cycle.min.js"></script>
<script type="text/javascript">
$(function() {
    $('#featured_slide').after('<div id="fs_pagination">').cycle({
        timeout: 5000,
        fx: 'fade',
        pager: '#fs_pagination',
        pause: 1,
```

```
        pauseOnPagerHover: 0
    });
});
</script>
</head>

<style>
    .inputs {
        background: #B2E1F5;
        font-size: 0.9rem;
        -moz-border-radius: 3px;
        -webkit-border-radius: 3px;
        border-radius: 3px;
        border: none;
        padding: 10px 10px;
        width: 200px;
        margin-bottom: 20px;
        box-shadow: inset 0 2px 3px rgba( 0, 0, 0, 0.1 );
        clear: both;
    }

    .inputs:focus {
        background: #fff;
        box-shadow: 0 0 0 3px #209ab2, inset 0 2px 3px rgba( 0, 0, 0, 0.2 ),
        0px 5px 5px rgba( 0, 0, 0, 0.15 );
        outline: none;
    }
}
```

```

    </style>
<body id="top">
<div class="wrapper col0">
    <div id="topline">
        <center> <h1 style="font-size: 22px">Sentiment Embeddings with
Applications to Sentiment Analysis </h1></center>

        <br class="clear" />
    </div>
</div>
<!--
#####
##### -->
<div class="wrapper col1">
    <div id="header">
        <div id="logo">
            <center>
                <h1><a
                    style="font-size: 30px"
href="index.html"><strong>S</strong>entiment
<strong>E</strong>mbeddings
<strong>A</strong>plications
<strong>S</strong>entiment
</a></h1></center>
                <strong>W</strong>ith
                <strong>T</strong>o
                <strong>A</strong>nalysis
            </div>
        </div>
    </div>
<div id="topnav">
    <ul>
        <li class="last"><a href="alog.jsp">Admin</a>
        <li><a href="#">User</a>

```

```

        <ul>
            <li><a href="uolog.jsp">Login</a></li>
            <li><a href="reg.jsp">Registration</a></li>
        </ul>
    </li>
<!--
-->    <li class=""><a href="index.jsp">Homepage</a></li>
        </ul>
    </div><

    <br class="clear" />
</div>

</div>

<!--
#####
##### -->

<div class="wrapper col2">
    <div id="featured_slide">
        <div class="featured_box">

             </div>
            <div class="featured_box">

                 </div>

```


<div class="featured_box">

 </div>

<div class="featured_box">

 </div>

</div>

</div>

<!--

-->

<!--<div class="wrapper col3">-->

<div class="container">

<div class="column">

<div class="subnav" style="float:right;margin-right: 0px">

<h2 style="font-size: 22px;font-style: italic">Side Menu Bar</h2>

<a style="font-size: 17px;color: royalblue"
href="ahome.jsp">Back

</div></div>

<div class="content">

<center>

```
<h2 style="color: royalblue;font-size: 20px">All
Products</h2><br><br><br>
```

```
</center><br><br><br>
```

```
<center><table align="right" style="text-align: center; margin-right:
100px; "> <tr>
```

```
<th style="text-align: center;width: 200px;color:
royalblue;font-size: 18px">Product Id</th>
```

```
<th style="text-align: center;width: 200px;color:
royalblue;font-size: 18px">Domain Name</th>
```

```
<th style="text-align: center;width: 200px;color:
royalblue;font-size: 18px">Product Name</th>
```

```
<th style="text-align: center;width: 200px;color:
royalblue;font-size: 18px">Price</th>
```

```
<th style="text-align: center;width: 200px;color:
royalblue;font-size: 18px">Description</th>
```

```
<th style="text-align: center;width: 200px;color:
royalblue;font-size: 18px">Product Image</th>
```

```
</tr>
```

```
<tr>
```

```
<%
```

```
Connection con = null;
```

```
Statement st = null;
```

```
ResultSet rs = null;
```

```
try {
```

```
con = DbConnection.getConnection();
```

```
st = con.createStatement();
```

```
rs = st.executeQuery("select * from newproduct");
```

```

        while (rs.next()) {
            %>

            <td><font                style="font-size:20px;color:
royalblue"><%=rs.getString("id")%></font></td>

            <td><font                style="font-size:20px;color:
royalblue"><%=rs.getString("domain")%></font></td>

            <td><font                style="color:        royalblue;font-size:
20px"><%=rs.getString("title")%></font></td>

            <td><font                style="color:        royalblue;font-size:
20px"><%=rs.getString("price")%></font></td>

            <td><font                style="color:        royalblue;font-size:
20px;"><%=rs.getString("discription")%></font></td>

            <td>" /></td>

        </tr>

        <% }

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

        %>

    </table>    </center>

    <!--    </div>-->

    <!--<div class="column" style="float:right;margin-right: -300px;margin-
top: 0px"-->    </div></div></body></html>

```

REFERENCES

- [1] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning Sentiment-specific word embedding for twitter sentiment classification," in Proc. 52th Annu. Meeting Assoc. Comput. Linguistics., 2014, pp. 1555–1565.
- [2] D. Tang, F. Wei, B. Qin, M. Zhou, and T. Liu, "Building large-scale twitter-specific sentiment lexicon: A representation learning approach," in Proc. 25th Int. Conf. Comput. Linguistics, 2014, pp. 172–182.
- [3] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, "Cooooo!!!: A deep learning system for twitter sentiment classification," in Proc. 8th Int. Workshop Semantic Eval., 2014, pp. 208–212.
- [4] C. D. Manning and H. Sch€utze, Foundations of Statistical Natural Language Processing. Cambridge, MA, USA: MIT Press, 1999.
- [5] D. Jurafsky and H. James, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Englewood Cliffs, NJ, USA: Prentice-Hall, 2000.
- [6] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," J. Mach. Learning Res., vol. 3, pp. 1137–1155, 2003.

- [7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. Conf. Neural Inf. Process. Syst., 2013, pp. 3111–3119.
- [8] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2014, pp. 1532–1543.
- [9] Z. S. Harris, “Distributional structure,” *Word*, vol. 10, pp. 146–162, 1954.
- [10] N. Yang, S. Liu, M. Li, M. Zhou, and N. Yu, “Word alignment modeling with context dependent deep neural network,” in Proc. 51st Annu. Meeting Assoc. Comput. Linguistics, 2013, pp. 166–175