|  |
| --- |
| **An analysis of model and feature-focused approaches of sentiment classification** |

**Kanica Ahuja Kushagradhi Bhowmik**

UIN: 675453362 UIN: 665435416

Department of Computer Science Department of Computer Science

University of Illinois at Chicago University of Illinois at Chicago

*kahuja4@uic.edu* [kbhowm2@uic..edu](mailto:kbhowm2@uic..edu)

**Md Atiqul Islam Mohitkumar Paritosh Ghia**

UIN: 673402596 UIN: 667342371

Department of Electrical and Engineering Department of Computer Science

University of Illinois at Chicago University of Illinois at Chicago

[msilam23@uic.edu](mailto:msilam23@uic.edu) [*mghia2@uic.edu*](mailto:mghia2@uic.edu)

**Nikhita Naramsetti Shibin Mathew**

UIN: 662265273 UIN: 656794467

Department of Computer Science Department of Computer Science

University of Illinois at Chicago University of Illinois at Chicago

[*snaram2@uic.edu*](mailto:snaram2@uic.edu)[*smathe36@uic.edu*](mailto:smathe36@uic.edu)

**Abstract**

We explore contemporary context-based approach for sentiment analysis and compare its performance with traditional machine learning techniques- Naive Bayes, Support Vector Machine (SVM), and Random Forest. The context-based approach identities key terms indicative of the existence of sentiment, which are used to generate language-based models to extract features for supervised learning. We pit the algorithms against each other on above mentioned datasets to evaluate their performance and illustrate their limitations in individual cases.

**1 Introduction**

Sentiment analysis refers to the inference of people’s views, positions and attitudes in their written or spoken texts. We used Context-based Sentiment analysis (ConSent), an approach for sentiment analysis effective both for regular texts (those that adhere to the rules of grammar and use existing words) and texts with a high degree of noise.

**1.1 Approach**

Our approach consists of two phases:

1. Apply techniques from the field of information retrieval to detect key terms in the text and analyze the context in which they appear.
2. Use the detected terms to generate features for supervised learning.

**1.2 Advantages**

Consent has several traits that make it very suitable for sentiment analysis in general and noisy text in particular:

1. Considers the context of terms, which is important when the text is noisy.
2. Does not rely on grammatical structures, which may not be reliable in noisy text.
3. Enables an easy integration of information from additional sources, such as the metadata of the analyzed text (e.g., the length of the text, the time of creation, the use of emoticons, etc.)

**2 Dataset**

We tested on two benchmark datasets:

1. Large Movie Review Dataset v1.0 [Maas et al., *Learning Word Vectors for Sentiment Analysis (ACL 2011)*] {1}

* 50,000 reviews split evenly into 25k training and 25k test sets.
* The overall distribution of labels is balanced (25k pos and 25k neg).
* Neutral reviews are not included - Negative review are movies with rating <=4, Positive review are movies with rating >=7 (on 10-point scale)
* No more than 30 reviews for any given movie because reviews for the same movie tend to have correlated ratings.

1. Twitter Dataset

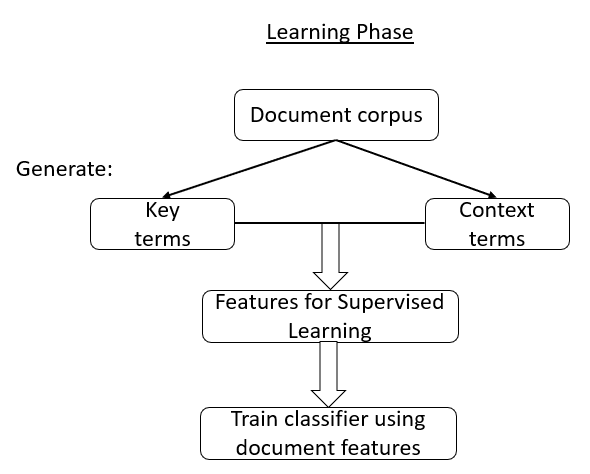
**3 Algorithm**

Consent consists of two phases: learning and detection.

**3.1 The Learning Phase**

The learning process consists of three steps:

1. Identify key terms – terms indicator of the sentiment.
2. Identify context term for each key term – to model text in key term vicinity
3. Use these terms to create features

Figure 1: Learning Phase

**3.1.1 Key Term Identification process**

1. Generate a language model for the set of positive-sentiment documents (Tpos). Denote this language model as lmpos.
2. Generate a language model for the set of negative-sentiment documents (Tneg). Denote this language model as lmneg.
3. For each term t in lmpos compute score (t), which is the ratio of its frequency in positive-sentiment and negative-sentiment documents.
4. If score (t) exceeds a predefined threshold, t is identified as a key term.

Algorithm 1: Identifying the key terms in the training set.

**Generate\_Key\_Terms**(*Tpos, Tneg, key\_term\_val\_threshold*)

1. *key\_terms list* ← ∅
2. *lmpos* ← **Generate\_Language\_Model**(Tpos)
3. *lmneg* ← **Generate\_Language\_Model**(Tneg)
4. for each (t in lmpos)
   1. *term\_indicativeness* = *lmpos(t)*/*lmneg(t)*
   2. if (*term indicativeness* ≥ *key\_term val\_threshold*)
   3. *key\_terms\_list*.Add(*t, term\_indicativeness*)
5. **return** *key\_terms\_list*

**3.1.2 Context Term Identification process**

1. Locate all instances of tkey both in Tpos and in Tneg.
2. For each instance found in T pos, extract the terms located around tkey by using a sliding window of size X denoted as the context span.
3. Denote excerpts from positive and negative as dpos and dneg, respectively. Dpos and Dneg are set of all dpos and dneg.
4. Generate language models for Dpos and Dneg and denote them lmDpos and lmDneg.
5. For each ‘‘candidate’’ context term tcontext in lmDpos we calculate its score using:  
    score(tcontext)= lmDpos(tcontext) - lmDneg (tcontext)
6. If the score of tcontext exceeds a predefined threshold, it will be defined as a context term for the key term tkey.

Algorithm 2: Identifying the context terms in the training set.

**Generate\_Context\_Terms(*Tpos*, *Tneg, key\_terms\_list*,**

***context\_term\_val\_threshold*, *context\_span*)**

1. *context\_terms\_list* ← ∅
2. foreach (*kt* in key\_terms\_list)
   1. *text\_sectionspos* ← **Generate\_Text\_Sections\_Containing\_KT**(*kt*, *Tpos*, *context\_span*)
   2. *text\_sectionsneg* ← **Generate\_Text\_Sections\_Containing\_KT**(kt, *Tneg*, *context\_span*)
   3. lmpos ← **Generate\_Language\_Model**(*text\_sectionspos*)
   4. lmneg ← **Generate\_Language\_Model**(*text\_sectionsneg*)
   5. foreach (*t* in lmpos)
      1. *termindicativeness* = lmpos(t)-lmneg (t)
      2. if (*term\_indicativeness* ≥ *context\_term\_val\_threshold*)
         1. *context\_terms\_list*.Add(*t, term\_indicativeness*)
3. **return** *context\_terms\_lis*t

**3.1.3 Creating Features**

Algorithm 3: Locating key and context terms in each document in

the training set and generating features.

**Locate\_Terms\_And\_Generate\_Features**(D, key\_terms\_list**,** context\_span)

1. Foreach (*d* ϵ *D*) //for each document in the documents set
   1. Foreach (*kt* ϵ *key\_terms\_list*)

1.1.1. If (! *d*.Contains(*kt*))

//if the key term does not appear in the document

* + - 1. **Continue**

//we now update the properties of each *kt* object

* + 1. *kt.positions* ← **Locate\_Key\_Term\_Positions\_In\_Text**(*d*,*kt*) //locate all the instances of the key term in the text and their positions
    2. *kt*.*context\_terms* ← **Locate\_Context\_Terms\_Positions\_In\_Text**(*d*, *kt.positions*,*kt.context\_terms*,*context\_span*)

//locate the context terms of the analyzed key term that also appear in the text

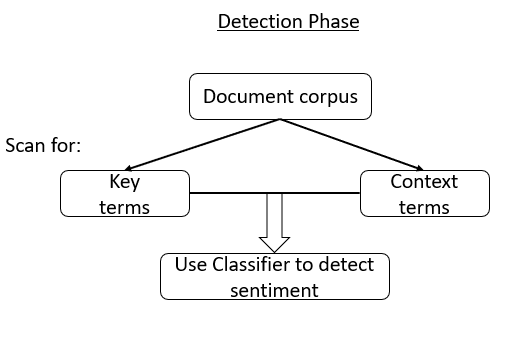
* 1. **Generate\_Features\_For\_Document**(*D*,*d*,*key\_terms\_list*)

//utilize the identified key and context terms to generate the features set for the document. Note that *key\_terms\_list* contains all the *kt* object, including those that were updated

**3.2 The Detection Phase**

In detection process takes place as follows:

1. text of the newly analyzed document is scanned for the key terms.
2. for each detected key term, we scan the text around it in search of its context terms.
3. Finally, using the key and context term we generate the features presented and use a classifier to produce a score for the document.

Figure 2: Detection Phase 

Algorithm 4: Locating key and context terms in a document

during the detection phase and calculating its score.

**Calculate\_Document\_Score**(*d*, *key\_terms\_list*, *context\_terms\_list*, *context\_span*, *classifier*)

1. Foreach (*kt* ϵ *key\_terms\_list*)
   1. If (!*d*.Contains(*kt*)) //if the key term does not appear in the document
      1. **Continue**

//we now update the properties of each kt object

* 1. *kt.positions* ← **Locate\_Key\_Term\_Positions\_In\_Text**(*d*,*kt*)
  2. *kt.context\_terms* ← **Locate\_Context\_Terms\_Positions\_In\_ Text**(*d*, *kt.positions*, *kt.context\_terms*, *context\_span*)

1. *document\_features* ← **Generate\_Features\_For\_Document**(*d*, *key\_terms\_list*) //Note that *key\_terms\_list* contains all the *kt* object, including those that were updated
2. *document\_score* ← **Classify\_Document**(*classifier*,*document\_features*) //here we call the machine learning-based classifier that utilizes the features generated for the analyzed document in step 2 in order to determine its score
3. **return** *document\_score*

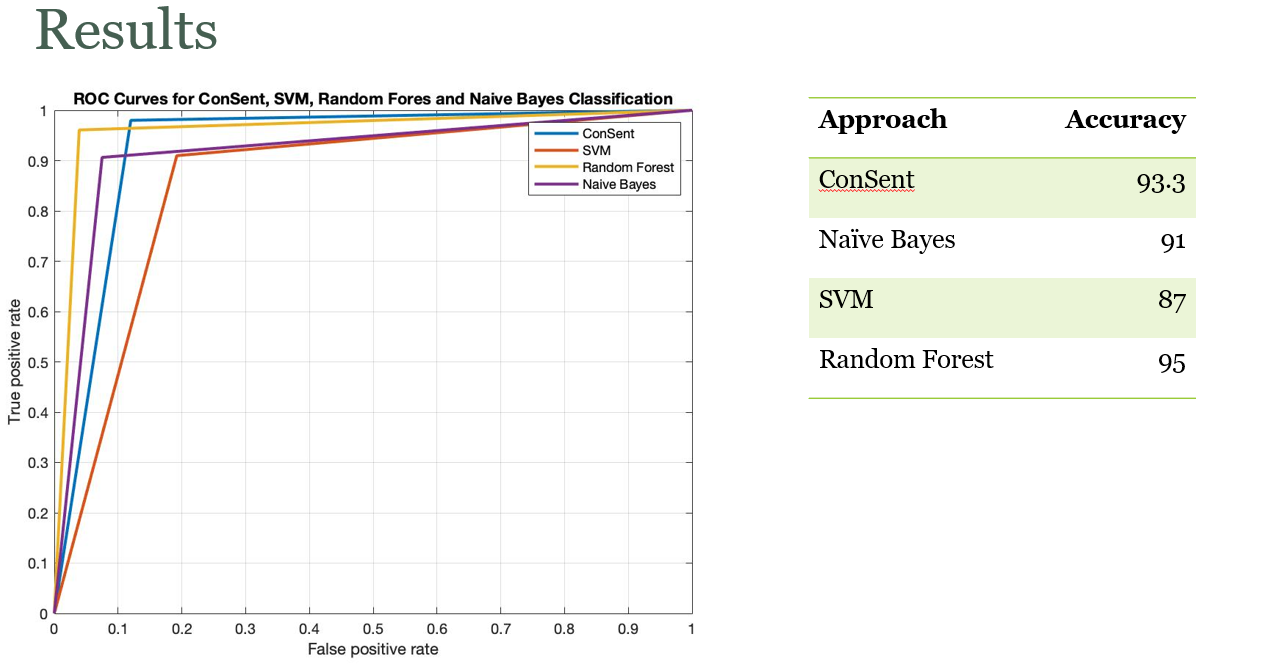
**4 Evaluating Measures**

We use two measures in order to evaluate the performance of the algorithms:

(a) Accuracy – this measure presents the overall percentage of correctly classified instances, regardless of their type. This measure is the one used to evaluate the performance of sentiment analysis.

(b) Area under the curve (AUC) – this measure calculates the area under the Receiver Operating Characteristic (ROC) curve.

**5 Result and Comparison**

Figure 3: Results

**6 References**