

SES: Sentiment Elicitation System for Social Media Data

Kunpeng Zhang, Yu Cheng, **Yusheng Xie**, Daniel Honbo, Ankit Agrawal, Diana Palsetia, Kathy Lee, Wei-keng Liao, Alok Choudhary

Dept. of Electrical Engineering and Computer Science
Center for Ultra-Scale Computing and Security
Northwestern University

{kzh980,ych133,yxi389,dkh301,ankitag,drp925,kml649,wkliao,choudhar}@eecs.northwestern.edu

**2011 IEEE International Conference on Data Mining
Vancouver, Canada, December 2011**

Introduction

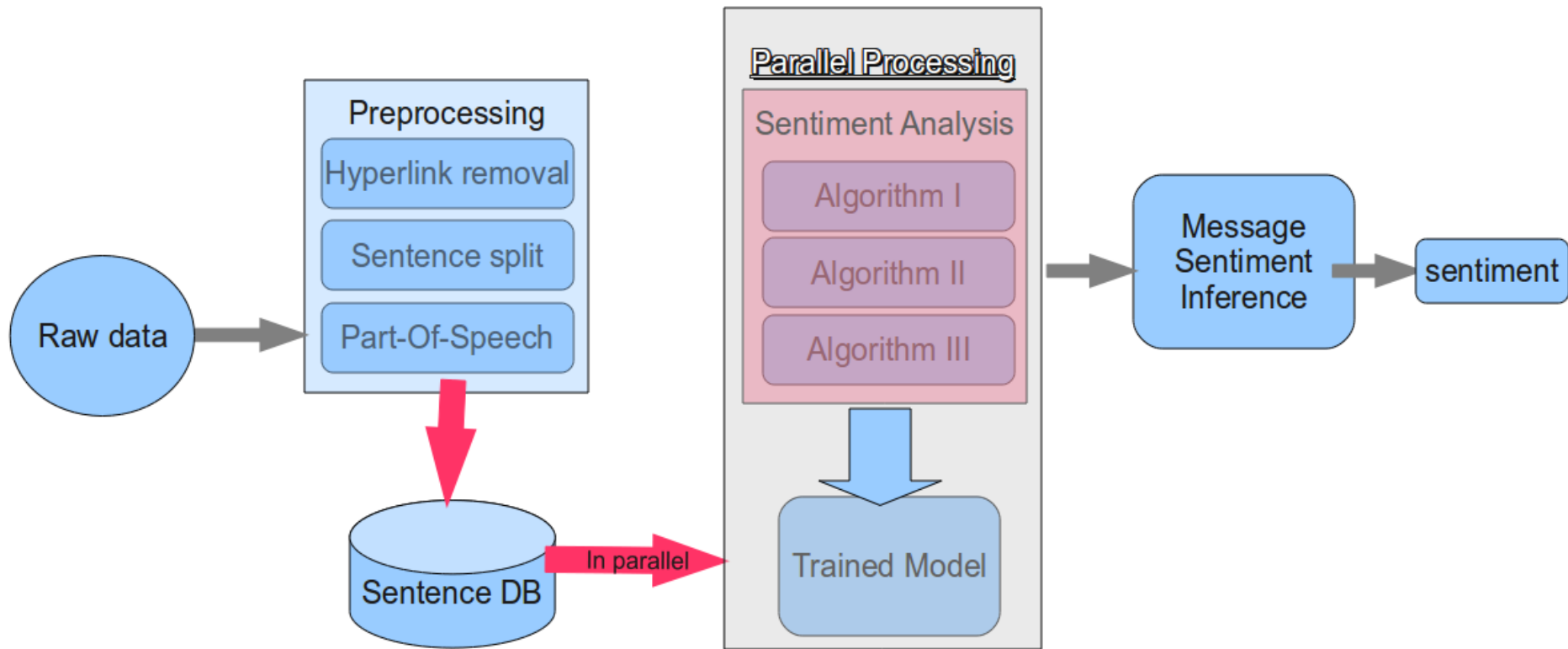
1. **Growth of social media data**
 - Facebook, LinkedIn, Twitter, Blogs, Customer reviews from online shopping websites, etc.
2. **Data analysis helps make informed decisions**
 - Social network analysis (community detection, friend recommendation, etc.)
 - Social media text analysis (review summarization, review sentiment identification)
3. **An ensemble method to identify sentiment of social media data**

Introduction(cont'd)

Problem description: given a sentence, we classify it into one of three categories(Positive, Negative, Neutral).

- Positive: directly or indirectly praise something, e.g. *“I love it! (^_^)”*
- Negative: directly or indirectly criticize something, e.g. *“We don’t like it at all. ☹️”*
- Neutral: No sentiments, or express a truth. e.g. *“The president should take that action.”*

System Description



SES Architecture

Text Pre-processing

1. Remove hyperlinks(*http://, https://, ftp://, etc.*)
2. Correct spellings(e.g. *luv=>love*)
3. Split long messages into sentences
4. POS: tag part-of-speech(*adj, adv, noun, verb, etc.*)

Individual Algorithms

Compositional semantic rules(CSR)

Table I: Compositional semantic rules

Rules	Example
1. Polarity(not [arg1]) = \neg polarity(arg1)	not [bad]{arg1}
2. Polarity([VP1] [NP1]) = Compose([VP], [NP])	[destroyed]{VP} the [terrorism]{NP}
3. Polarity([VP1]to[VP2]) = Compose([VP1],[VP2])	[refused]{VP1} to{to} [deceive]{VP2} the man
4. Polarity([ADJ]to[VP1]) = Compose([ADJ],[VP1])	unlikely{ADJ} to{to} [destroy]{VP} the planet
5. Polarity([NP1]in[NP1]) = Compose([NP1],[NP2])	[lack]{NP1} offing [crime]{NP2} in rural areas
6. Polarity([NP1] [VP1]) = Compose([VP1],[NP1])	crime{NP1} has [decreased]{VP1}
7. Polarity([NP1]be[ADJ]) = Compose([ADJ],[NP1])	[damage]{NP1} is{be} [minimal]{ADJ}
Our Rules	Example
8. Polarity([NP1]in[VP1]) = Compose([NP1],[VP1])	lack{NP1} offing killing{VP1} in rural areas
9. Polarity(as[ADJ]as[NP]) = if(polarity(NP) !=0: return polarity(NP), else: return polarity(ADJ)	as{as} ugly{ADJ} as{as} a rock{NP}
10. Polarity(not as[ADJ]as[NP]) = -polarity(ADJ)	that was not{not} as{as} [bad]{ADJ1} as the original]{NP2}
11. If the sentence contains 'but', disregard all previous sentiment only take the sentiment of the sentence after 'but'	and I've never liked that director, [but] I loved this movie.
12. If the sentence contains 'despite', only the sentiment in the previous part of the sentence is counted.	I love that movie, despite the fact that I hate the director.

CSR(cont'd)

Table II: The compose function is used to derive the polarity of an expression. The table lists the Compose function used in [1]. Compose2 is our version of the Compose function

Compose(arg1,arg2)	if(arg1 is a negator then \neg polarity(arg2) else if (Polarity(arg1) == Polarity(arg2)) then Polarity(arg1) else the majority polarity of data
Compose2(arg1,arg2)	if arg1 is negative: if arg2 is not neutral: return : polarity(arg2) else: return -1 else if arg1 is positive and arg2 is not neutral: return polarity(arg2) else if polarity(arg1) equals polarity(arg2): return 2 polarity(arg1) else if (arg1 is positive and arg2 is neutral) or (arg2 is positive and arg1 is neutral): return polarity(arg1) + polarity(arg2) else: return 0

Individual Algorithms(cont'd)

Numeric sentiment identification(NSI)

Strength of sentiment words: *the sentiment scores are not only associated with user star ratings but also word appearance frequency.*

$$Score(w) = \frac{\sum_{i \in P} \frac{n_{r_i}}{N} r_i f_{r_i}}{\sum_{r_i} \frac{n_{r_i}}{N} f_{r_i}}$$

$$Score(adv) = Pol(adj) \cdot (Score(adv, adj) - Score(adj))$$

adjective words			
<i>Adj</i>	<i>score</i>	<i>Adj</i>	<i>score</i>
Easy	4.1	hard	2.5
Good	3.9	Best	5.0
Rude	1.69	Worst	1.0
Nice	3.7		
Bad	1.5		
Great	4.5		

adverb words			
<i>Adv</i>	<i>score</i>	<i>Adv</i>	<i>score</i>
Super	0.41	Pretty	0.18
Not	-1.90	Most	1.0
Little	-0.16	Never	-2.0
A bit	0.03		
Pretty	0.06		
Really	0.42		

NSI(cont'd)

Derivation of sentence score:

$$\text{Score}(\text{adv}(\text{adj}(\text{noun}))) = \text{Score}(\text{adj}) + \text{Pol}(\text{adj}) \cdot \text{Score}(\text{adv})$$

$$\text{Score}(\text{adv}(\text{verb})) = \text{Score}(\text{verb}) + \text{Pol}(\text{verb}) \cdot \text{Score}(\text{adv})$$

$$\begin{aligned} \text{Score}(\text{neg}(\text{adv}(\text{adj}))) &= \text{Score}(\text{adj}) + \text{Pol}(\text{adj}) \cdot \text{Score}(\text{adv}) \\ &\quad + \text{Pol}(\text{adj}) \cdot \text{Score}(\text{neg}) \end{aligned}$$

$$\begin{aligned} \text{Score}(\text{neg}(\text{adv}(\text{verb}))) &= \text{Score}(\text{verb}) + \text{Pol}(\text{verb}) \cdot \text{Score}(\text{adv}) \\ &\quad + \text{Pol}(\text{verb}) \cdot \text{Score}(\text{neg}) \end{aligned}$$

Algorithm 1 Calculating numeric score for a sentence

1. assign scores to all adjectives and adverbs
 2. extract all phrases(P) and calculate each score(PS)
 3. $S = \sum_{i=1}^m PS_i$, where m is the number of phrases
-

Individual Algorithms(Cont'd)

Bag-of-word & Rule-based

◆ Emoticons

- ◆ 77 positive emoticons :)
- ◆ 59 negative emoticons :(

◆ Internet language

- ◆ [Thank you | go], [a company(organization) name | a person name] : *1st comment! Go Scuderia! Ferrari Go!*

◆ Domain specific keywords

- ◆ *“Yum, Yummy” should be positive word for food comments.*

Machine Learning Model

- ◆ Ensemble learning
- ◆ 3 basic features
 - ◆ Score from CSR algorithm: $S1$ $[-10, +10]$
 - ◆ Score from NSI algorithm: $S2$ $[-5, +5]$
 - ◆ Sentiment from the third algorithm: $S3$ $\{P, N, O\}$
- ◆ 2 derived features
 - ◆ $S1 + S2$
 - ◆ $S2 - S1$

Machine Learning Model (cont'd)

- Predictive models
 - ◆ Decision tree
 - ◆ Neural network
 - ◆ Logistic regression
 - ◆ Random forest

Inference of Message Sentiment

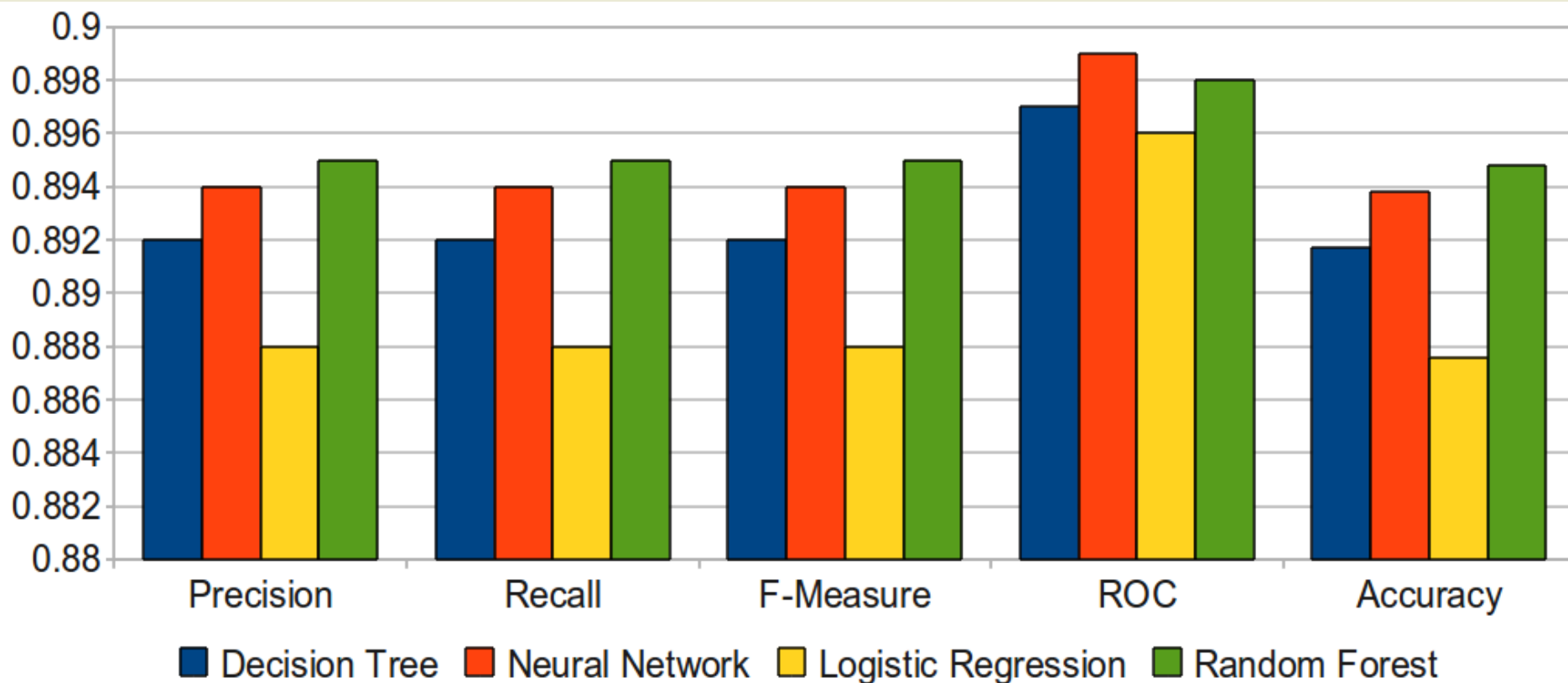
It's common that users use both positive and negative sentences to express their opinions within one comment

5 categories

- ◆ Positive: *All sentences are positive and subjective*
- ◆ Negative: *All sentences are negative and subjective*
- ◆ Positive mixed: *The number of positive sentences is greater than the number of negative sentences(>0)*
- ◆ Negative mixed: *The number of negative sentences is greater than the number of positive sentences(>0)*
- ◆ Mixed: *The number of negative sentences is equal to the number of positive sentences*

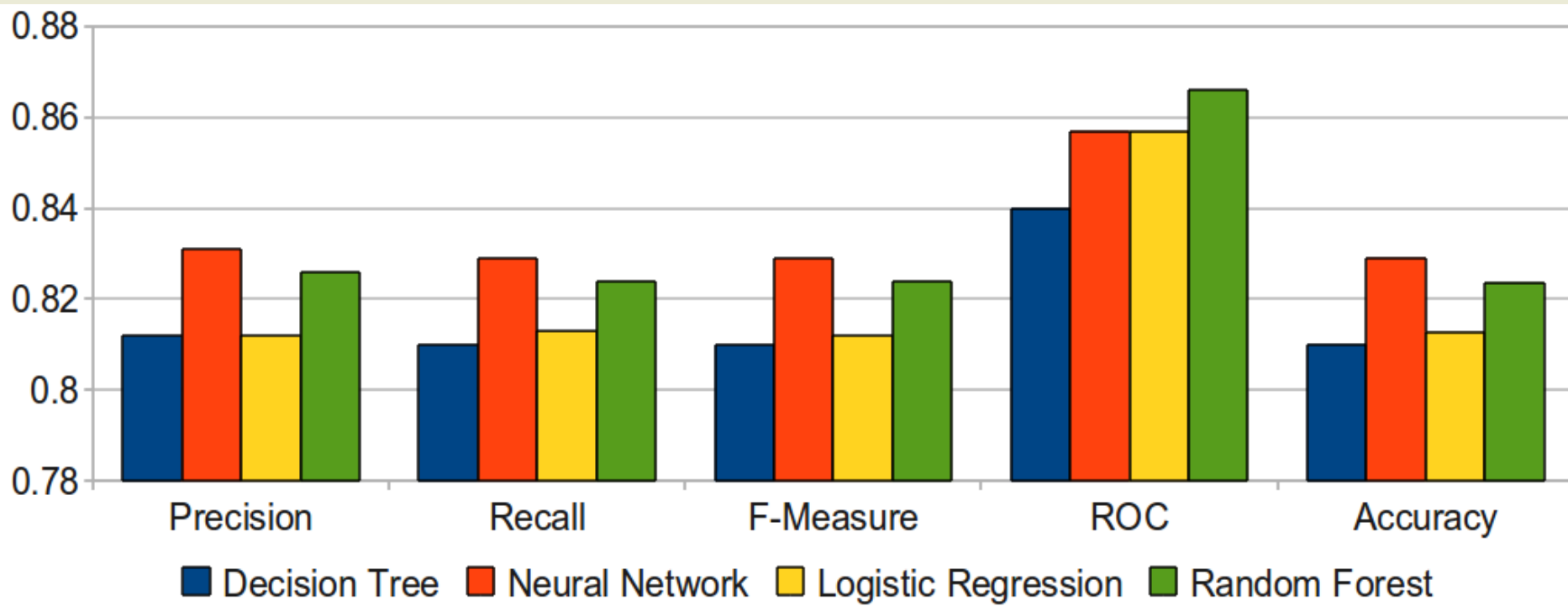
Experiment Results

2000 Facebook comments



Experiment Results (cont.)

1000 Twitter tweets



Related Work

1. **Sentiment analysis** [*B. Liu, 2010; B. Pang, 2002*]
2. **Extracting product features** [*M. Hu, 2004; A. Popescu, 2005*]
3. **Review summarization** [*M. Hu, 2004, 2006*]

Conclusions & Future Work

- ◆ Expand compositional semantic rules and numerical sentiment identification phrases.
- ◆ Ensemble three individual algorithms to classify sentence sentiments.
- ◆ Incorporate “mixed” concept to label the message sentiment.
- ◆ Future improvements
 - ◆ Capture contextual information
 - ◆ Include active learning to improve learning

Thank You

**Dept. of Electrical Engineering and Computer Science
Center for Ultra-Scale Computing and Security
Northwestern University**

**2011 IEEE International Conference on Data Mining
Vancouver, Canada, December 2011**