TwiSent: A Multi-Stage System for Analyzing Sentiment in Twitter

Subhabrata Mukherjee, Akshat Malu, Balamurali A.R.
and Pushpak Bhattacharyya
Dept. of Computer Science and Engineering, IIT Bombay

21st ACM Conference on Information and Knowledge Management
CIKM 2012,
Hawai, Oct 29 - Nov 2, 2012

Social Media Analysis

Had Hella fun today with the team. Y'all are hilarious! &Yes, i do need more black homies......

Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily

Had Hella fun today with the team. Y'all are hilarious! &Yes, i do need more black homies......

Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily

- This information content could be leveraged to create applications that have a social as well as an economic value

Had Hella fun today with the team. Y'all are hilarious! &Yes, i do need more black homies......

Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily
- This information content could be leveraged to create applications that have a social as well as an economic value
- Text limit of 140 characters per tweet makes Twitter a noisy medium
  - Tweets have a poor syntactic and semantic structure
  - Problems like *slangs, ellipses, nonstandard vocabulary etc.*

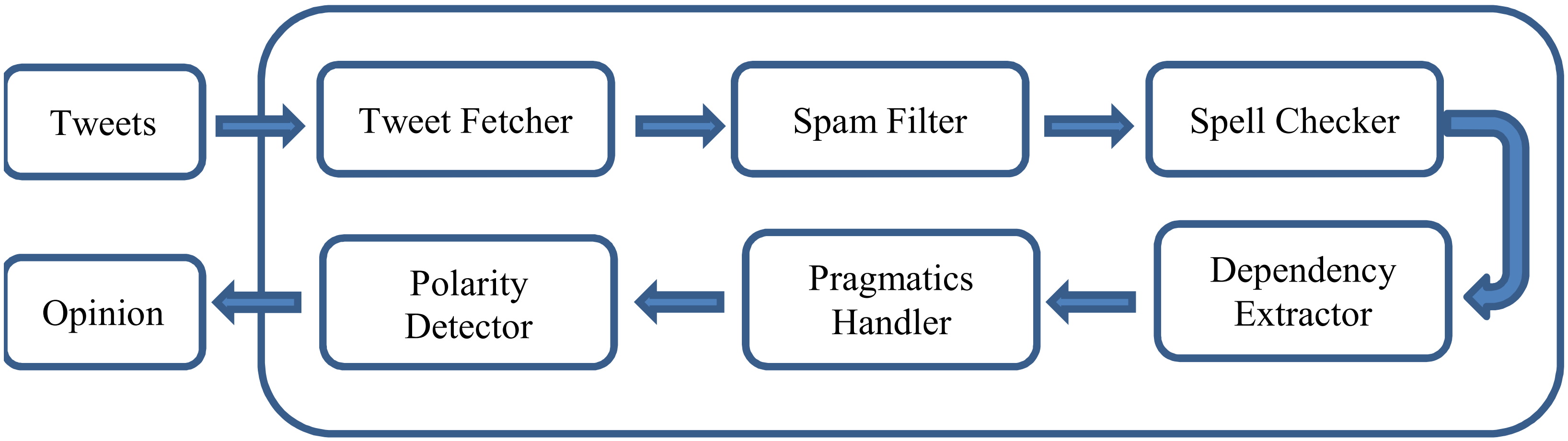
Had Hella fun today with the team. Y'all are hilarious! &Yes, i do need more black homies......

Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily
- This information content could be leveraged to create applications that have a social as well as an economic value
- Text limit of 140 characters per tweet makes Twitter a noisy medium
  - Tweets have a poor syntactic and semantic structure
  - Problems like *slangs, ellipses, nonstandard vocabulary etc.*
- Problem is compounded by increasing number of *spams* in Twitter
  - P*romotional* tweets, *bot-generated* tweets, *random links* to websites etc. In fact Twitter contains around 40% tweets as pointless babble

Had Hella fun today with the team. Y'all are hilarious! &Yes, i do need more black homies......

TwiSent: Multi-Stage System Architecture



Spam Categorization and Features

- *Re-tweets*
- *Promotional tweets for some entity*
- *Tweets containing links to some other websites*
- *Tweets in languages other than English*
- *Tweets with incomplete text*
- *Automatically generated tweets by bots*
- *Tweets built primarily for search engines or tweets with excessive off-topic keywords*
- *Multiple tweets offering substantially the same content*

1. *Number of Words per Tweet*	**9.** *Frequency of Foreign Words*
	10. *Validity of First Word*
2. *Average Word Length*	**11.** *Presence / Absence of links*
3. *Frequency of "?" and "!"*	**12.** *Frequency of POS Tags*
	13. *Strength of Character Elongation*
4. *Frequency of Numeral Characters*	**14.** *Frequency of Slang Words*
	15. *Average Positive and Negative Sentiment of Tweets*
5. *Frequency of hashtags*	
6. *Frequency of @users*	
7. *Extent of Capitalization*	
8. *Frequency of the First POS Tag*	

Algorithm for Spam Filter

Input: Build an initial naive bayes classifier NB- C, using the tweet sets *M* (mixed unlabeled set containing spams and non-spams) and *P* (labeled non-spam set)

1: Loop while classifier parameters change

2: for each tweet $t_i \in M$ do

3: Compute $\Pr[c_1 | t_i]$, $\Pr[c_2 | t_i]$ using the current NB //c_1 - non-spam class , c_2 - spam class

4: $\Pr[c_2 | t_i] = 1 - \Pr[c_1 | t_i]$

5: Update $\Pr[f_{i,k} | c_1]$ and $\Pr[c_1]$ given the probabilistically assigned class for all t_i ($\Pr[c_1 | t_i]$). (a new NB-C is being built in the process)

6: end for

7: end loop

$$\Pr[c_j \mid t_i] = \frac{\Pr[c_j] \prod_k \Pr[f_{i,k} | c_j]}{\sum_r \Pr[c_r] \prod_k P(f_{i,k} | c_r)}$$

Categorization of Noisy Text

- Dropping of Vowels - *btfl (beautiful), lvng (loving)*
- Vowel Exchange - *good vs. gud (o,u)*
- Mis-spelt words - *redicule (ridicule), magnificant (magnificent)*
- Text Compression - *shok (shock), terorism (terrorism)*
- Phonetic Transformation - *be8r (better), gud (good), fy9 (fine), gr8 (great)*
- Normalization and Pragmatics - *hapyyyyyy (happy), guuuuud (good)*
- Segmentation with Punctuation - *beautiful, (beautiful)*
- Segmentation with Compound Words - *breathtaking (breath-taking), eyecatching (eye-catching), good-looking (good looking)*
- Hashtags and Segmentation - *#notevenkidding, #worthawatch*
- Combination of all - *#awsummm (awesome), gr88888 (great), amzng,btfl (amazing, beautiful).*

Spell-Checker Algorithm

Spell-Checker Algorithm

- Heuristically driven to resolve the *identified errors* with a *minimum edit distance based spell checker*

Spell-Checker Algorithm

- Heuristically driven to resolve the *identified errors* with a *minimum edit distance based spell checker*

- A *normalize* function takes care of *Pragmatics* and *Number Homophones*
  - Replaces *happppyyyy* with *hapy*, *‘2’ with ‘to’, ‘8’ with ‘eat’, ‘9’ with ‘ine’*

Spell-Checker Algorithm

- Heuristically driven to resolve the *identified errors* with a *minimum edit distance based spell checker*

- A *normalize* function takes care of *Pragmatics* and *Number Homophones*
  - Replaces *happpyyyy* with *hapy*, *‘2’ with ‘to’, ‘8’ with ‘eat’, ‘9’ with ‘ine’*

- A *vowel_dropped* function takes care of the *vowel dropping* phenomenon

Spell-Checker Algorithm

- Heuristically driven to resolve the *identified errors* with a *minimum edit distance based spell checker*

- A *normalize* function takes care of *Pragmatics* and *Number Homophones*
  - Replaces *happpyyyy* with *hapy*, *‘2’ with ‘to’, ‘8’ with ‘eat’, ‘9’ with ‘ine’*

- A *vowel_dropped* function takes care of the *vowel dropping* phenomenon

- The parameters *offset* and *adv* are determined empirically

Spell-Checker Algorithm

- Heuristically driven to resolve the *identified errors* with a *minimum edit distance based spell checker*

- A *normalize* function takes care of *Pragmatics* and *Number Homophones*
  - Replaces *happpyyyy* with *hapy*, *'2' with 'to', '8' with 'eat', '9' with 'ine'*

- A *vowel_dropped* function takes care of the *vowel dropping* phenomenon

- The parameters *offset* and *adv* are determined empirically

- Words are marked during normalization, to preserve their pragmatics

  happppyyyyy, normalized to *hapy* and thereafter spell-corrected to *happy,* is marked so as to not lose its pragmatic content

Spell-Checker Algorithm

- **Input:** For string s, let *S* be the set of words in the lexicon starting with the initial letter of *s*.
- /* Module Spell Checker */
- **for** each word *w* ∈ *S* **do**
-   *w'*=vowel_dropped(*w*)
-   *s'*=normalize(*s*)
- */*diff(s,w) gives difference of length between s and w*/*
-   **if** diff(*s'* , *w'*) < offset **then**
-     score[*w*]=min(edit_distance(*s*,*w*),edit_distance(*s*,
-     *w'*), edit_distance(*s'* , *w*))
-   else
-     score[*w*]=max_centinel
-   **end if**
- **end for**

Spell-Checker Algorithm Contd..

- Sort score of each *w* in the Lexicon and retain the top m entries in suggestions(*s*) for the original string *s*
- **for** each *t* in suggestions(*s*) **do**
- $edit_1$=edit_distance(*s'* , *s*)
- */*t.replace(char1,char2) replaces all occurrences of char1 in the string t with char2*/*
- $edit_2$=edit_distance(t.replace(*a* , *e*), *s'*)
- $edit_3$=edit_distance(t.replace(*e* , *a*), *s'*)
- $edit_4$=edit_distance(t.replace(*o* , *u*), *s'*)
- $edit_5$=edit_distance(t.replace(*u* , *o*), *s'*)
- $edit_6$=edit_distance(t.replace(*i* , *e*), *s'*)
- $edit_7$=edit_distance(t.replace(*e*, *i*), *s'*)
- count=overlapping_characters(*t, s'*)
- min_edit=
- min($edit_1$,$edit_2$,$edit_3$,$edit_4$,$edit_5$,$edit_6$,$edit_7$)
- **if** (min_edit ==0 or score[*s'*] == 0) **then**
- adv=-2 */* for exact match assign advantage score */*
- else
- adv=0
- **end if**
- final_score[t]=min_edit+adv+score[*w*]-count;
- **end for**
- return *t* with minimum final_score;

Feature Specific Tweet Analysis

- *I have an **ipod** and it is a <u>great</u> buy but I'm probably the only person that <u>dislikes</u> the iTunes **software**.*
  - *Here the sentiment w.r.t ipod is positive whereas that respect to software is negative*

Opinion Extraction Hypothesis

- *“More closely related words come together to express an opinion about a feature”*

Hypothesis Example

- *"I want to use Samsung which is a great product but am not so sure about using Nokia".*
  - *Here "great" and "product" are related by an adjective modifier relation, "product" and "Samsung" are related by a relative clause modifier relation. Thus "great" and "Samsung" are transitively related.*
  - ***Here "great" and "product" are more related to Samsung than they are to Nokia***
  - *Hence "great" and "product" come together to express an opinion about the entity "Samsung" than about the entity "Nokia"*

Hypothesis Example

- *"I want to use Samsung which is a great product but am not so sure about using Nokia".*
  - *Here "great" and "product" are related by an adjective modifier relation, "product" and "Samsung" are related by a relative clause modifier relation. Thus "great" and "Samsung" are transitively related.*
  - ***Here "great" and "product" are more related to Samsung than they are to Nokia***
  - *Hence "great" and "product" come together to express an opinion about the entity "Samsung" than about the entity "Nokia"*

Hypothesis Example

- *"I want to use Samsung which is a great product but am not so sure about using Nokia".*
  - *Here "great" and "product" are related by an adjective modifier relation, "product" and "Samsung" are related by a relative clause modifier relation. Thus "great" and "Samsung" are transitively related.*
  - ***Here "great" and "product" are more related to Samsung than they are to Nokia***
  - *Hence "great" and "product" come together to express an opinion about the entity "Samsung" than about the entity "Nokia"*

Hypothesis Example

- *"I want to use Samsung which is a great product but am not so sure about using Nokia".*
  - *Here "great" and "product" are related by an adjective modifier relation, "product" and "Samsung" are related by a relative clause modifier relation. Thus "great" and "Samsung" are transitively related.*
  - ***Here "great" and "product" are more related to Samsung than they are to Nokia***
  - *Hence "great" and "product" come together to express an opinion about the entity "Samsung" than about the entity "Nokia"*

Hypothesis Example

- *"I want to use Samsung which is a great product but am not so sure about using Nokia".*

Adjective Modifier

  - *Here "great" and "product" are related by an adjective modifier relation, "product" and "Samsung" are related by a relative clause modifier relation. Thus "great" and "Samsung" are transitively related.*
  - ***Here "great" and "product" are more related to Samsung than they are to Nokia***
  - *Hence "great" and "product" come together to express an opinion about the entity "Samsung" than about the entity "Nokia"*

Hypothesis Example

- *"I want to use Samsung which is a great product but am not so sure about using Nokia".*

Adjective Modifier

  - *Here "great" and "product" are related by an adjective modifier relation, "product" and "Samsung" are related by a relative clause modifier relation. Thus "great" and "Samsung" are transitively related.*
  - ***Here "great" and "product" are more related to Samsung than they are to Nokia***
  - *Hence "great" and "product" come together to express an opinion about the entity "Samsung" than about the entity "Nokia"*

Hypothesis Example

- *"I want to use Samsung which is a great product but am not so sure about using Nokia".*

Adjective Modifier

  - *Here "great" and "product" are related by an adjective modifier relation, "product" and "Samsung" are related by a relative clause modifier relation. Thus "great" and "Samsung" are transitively related.*
  - ***Here "great" and "product" are more related to Samsung than they are to Nokia***
  - *Hence "great" and "product" come together to express an opinion about the entity "Samsung" than about the entity "Nokia"*

Hypothesis Example

- *“I want to use Samsung which is a great product but am not so sure about using Nokia”.*

Relative Clause Modifier

Adjective Modifier

  - *Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.*
  - ***Here “great” and “product” are more related to Samsung than they are to Nokia***
  - *Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”*

Example of a Review

- *I have an **ipod** and it is a great buy but I'm probably the only person that dislikes the iTunes **software**.*

Example of a Review

- *I have an **ipod** and it is a great buy but I'm probably the only person that dislikes the iTunes **software**.*

Example of a Review

- *I have an **ipod** and it is a great buy but I'm probably the only person that dislikes the iTunes **software**.*

Example of a Review

- *I have an **ipod** and it is a great buy but I'm probably the only person that dislikes the iTunes **software**.*

Example of a Review

- *I have an **ipod** and it is a great buy but I'm probably the only person that dislikes the iTunes **software**.*

Feature Extraction : Domain Info Not Available

Feature Extraction : Domain Info Not Available

- Initially, all the Nouns are treated as features and added to the *feature list* ***F***.

Feature Extraction : Domain Info Not Available

- Initially, all the Nouns are treated as features and added to the *feature list* ***F***.

- ***F*** = *{ ipod, buy, person, software }*

Feature Extraction : Domain Info Not Available

- Initially, all the Nouns are treated as features and added to the *feature list* ***F***.

- ***F*** = *{ ipod, buy, person, software }*

- Pruning the feature set
  - Merge 2 features if they are **strongly related**

Feature Extraction : Domain Info Not Available

- Initially, all the Nouns are treated as features and added to the *feature list* ***F***.

- ***F*** = *{ ipod, buy, person, software }*

- Pruning the feature set
  - Merge 2 features if they are **strongly related**

- *"buy"* merged with "*ipod*", when target feature = "*ipod*",
  - *"person, software"* will be ignored.

Feature Extraction : Domain Info Not Available

- Initially, all the Nouns are treated as features and added to the *feature list* ***F***.

- ***F*** = *{ ipod, buy, person, software }*

- Pruning the feature set
  - Merge 2 features if they are **strongly related**

- *"buy"* merged with "*ipod", when* target feature = "*ipod",*
  - *"person, software"* will be ignored.

- *"person"* merged with "*software", when* target feature = "*software"*
  - *"ipod, buy"* will be ignored.

Relations

- Direct Neighbor Relation
  - Capture **short range dependencies**
  - Any 2 consecutive words (such that none of them is a StopWord) are directly related
  - Consider a sentence *S* and 2 consecutive words $w_i, w_{i+1} \in S$.
  - *If* $w_i, w_{i+1} \notin Stopwords$, then they are directly related.

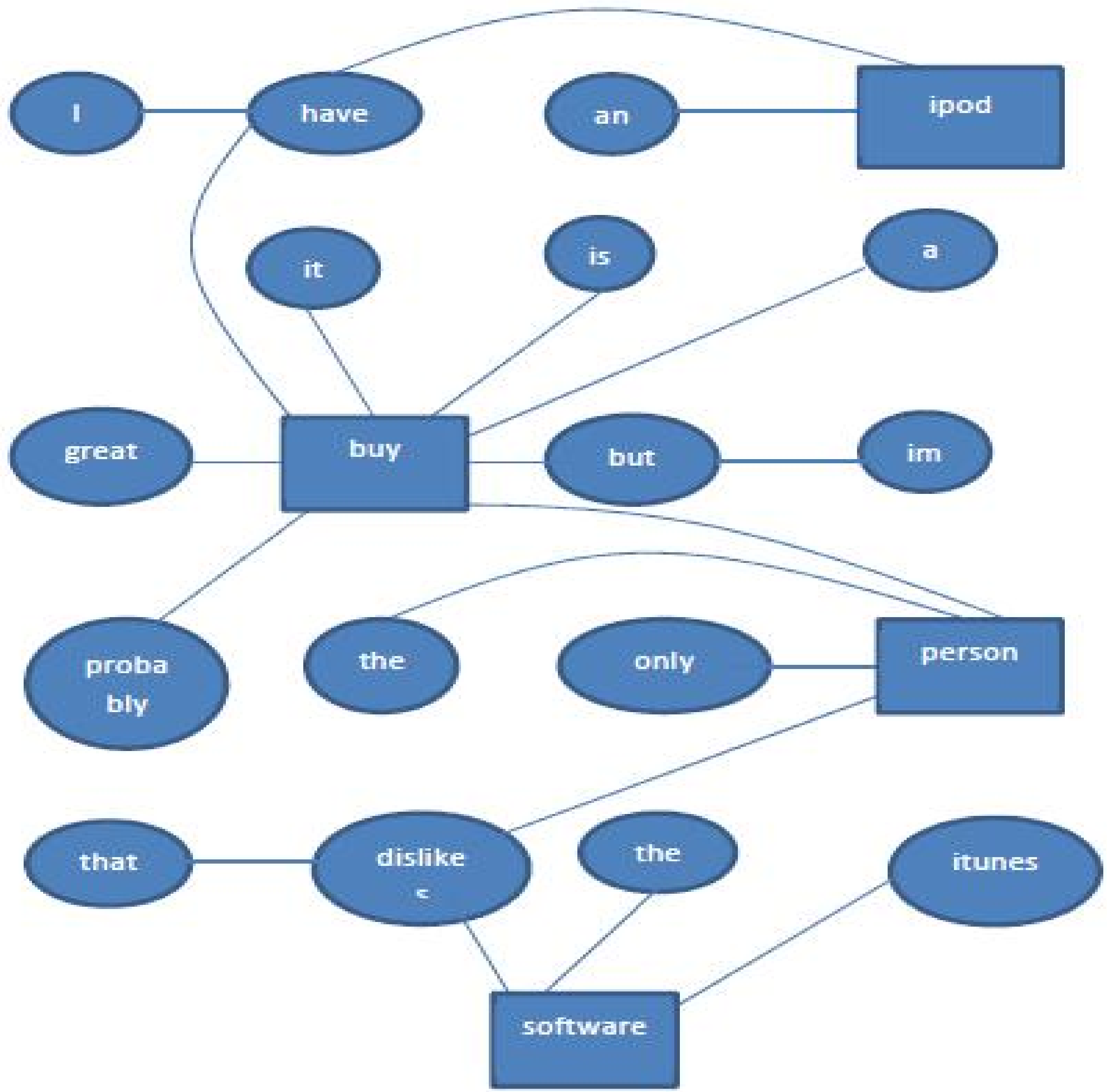
- Dependency Relation
  - Capture **long range dependencies**
  - Let *Dependency_Relation* be the list of **significant relations**.
- Any 2 words w_i and w_j in *S* are directly related, *if* $\exists D_i$ s.t. $D_i(w_i, w_j) \in Dependency\_Relation$

Graph representation

Given a sentence S, let $\boldsymbol{W}$ be the set of all words in the sentence S.

A Graph $G(W, E)$ is constructed such that any $w_i, w_j \in W$ are directly connected by $e_k \in E$, if $\exists R_l$ s.t. $R_l(w_i, w_j) \in R$.

Graph



I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

Algorithm

i. Initialize n clusters $C_i \ \forall i = 1..n$

ii. Make each $f_i \in F$ the clusterhead of C_i . The target feature f_t is the clusterhead of C_t. Initially, each cluster consists only of the clusterhead.

Algorithm Contd…

iii. Assign each word $w_j \in S$ to cluster C_k

$$s.t.\ \ k = \arg\min_{i \in n} dist(w_j, f_i),$$

Where $dist(w_j, f_i)$ *gives the number of edges, in the shortest path, connecting* w_j *and* f_i *in G.*

Algorithm Contd…

iv. Merge any cluster C_i with C_t *if*,

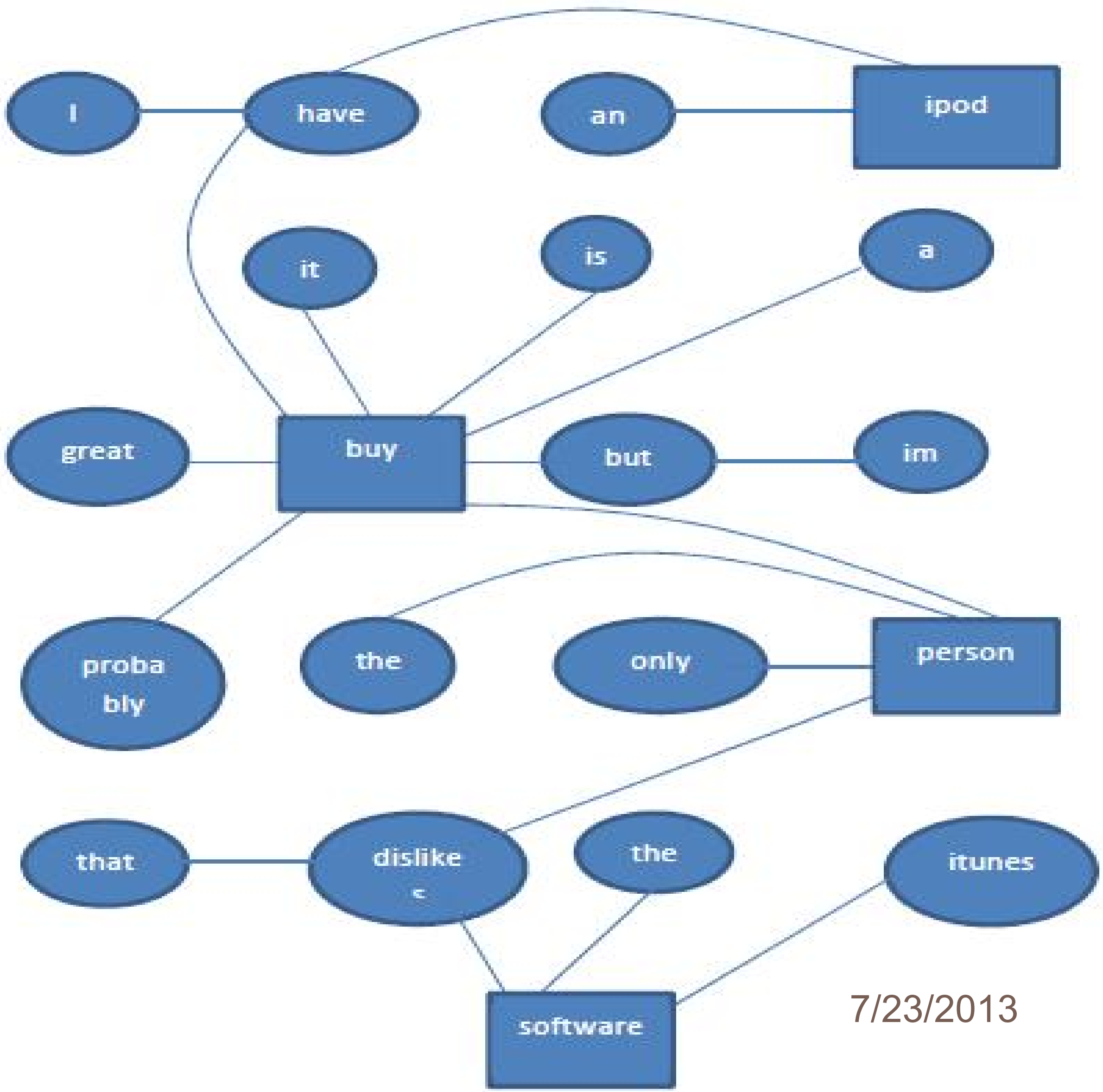
$$dist(f_i, f_t) < \theta \,,$$

Where θ is some threshold distance.

v. Finally the set of words $w_i \in C_t$ gives the opinion expression regarding the target feature f_t .

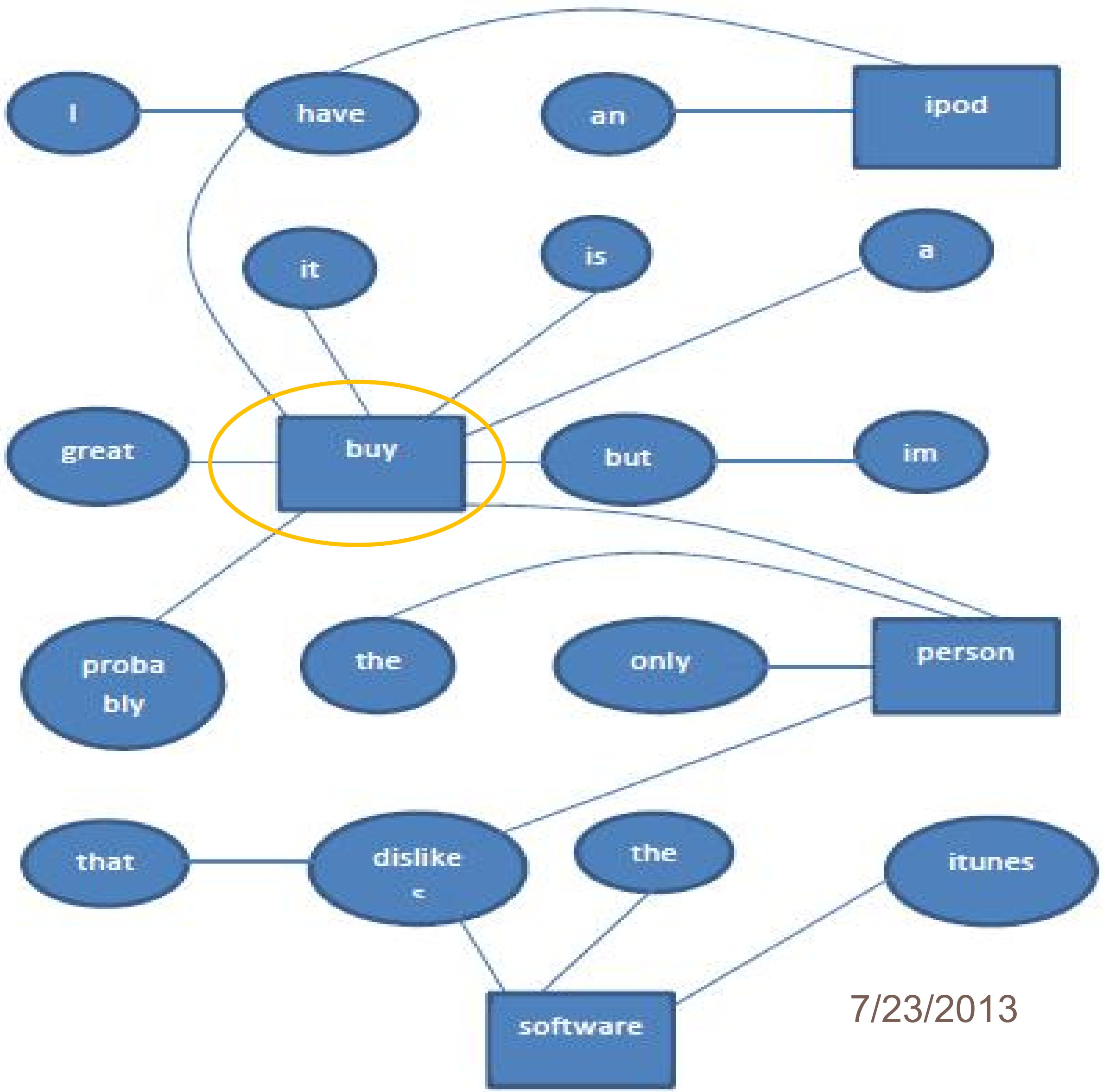
Clustering

Clustering

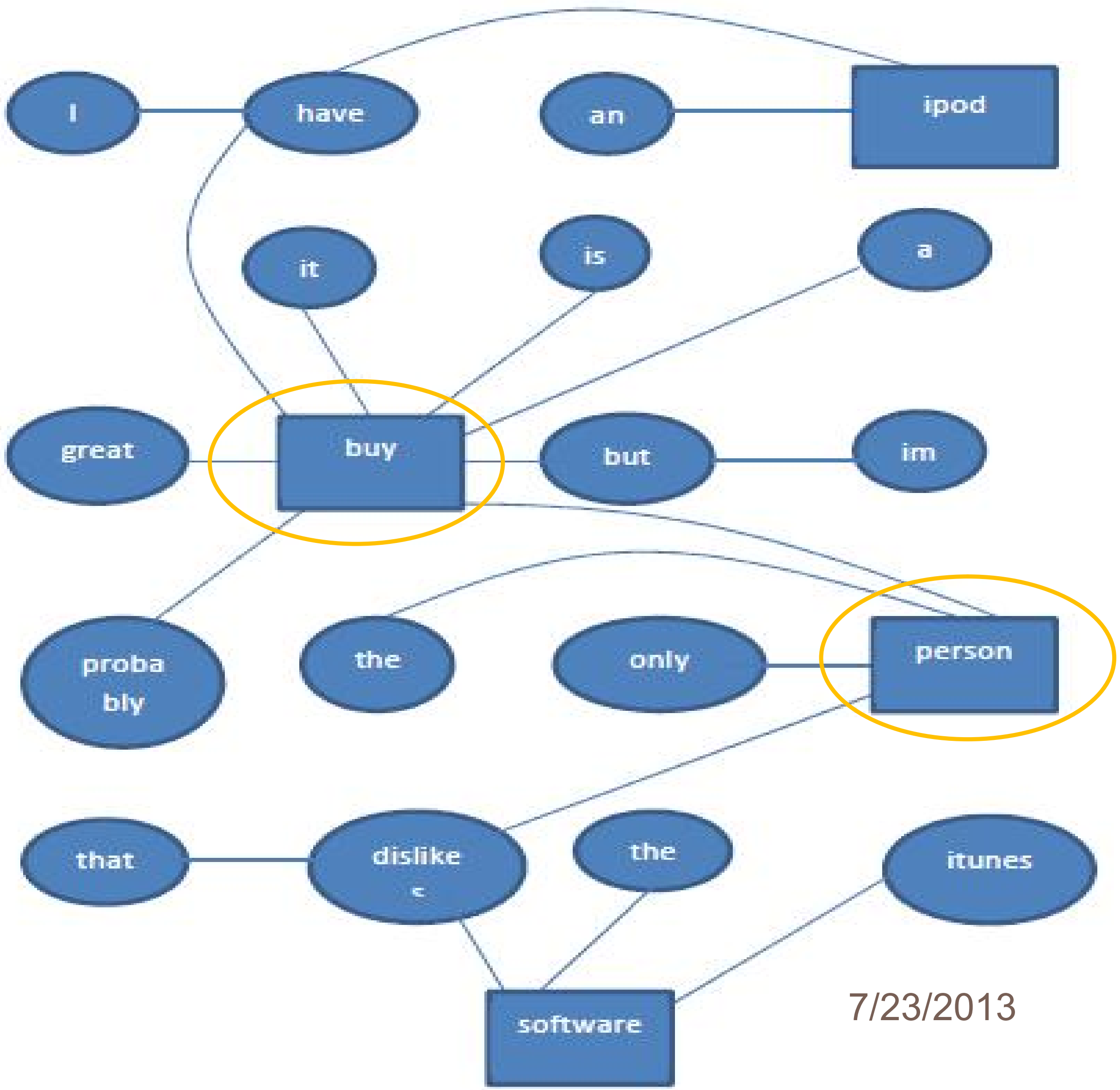


I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

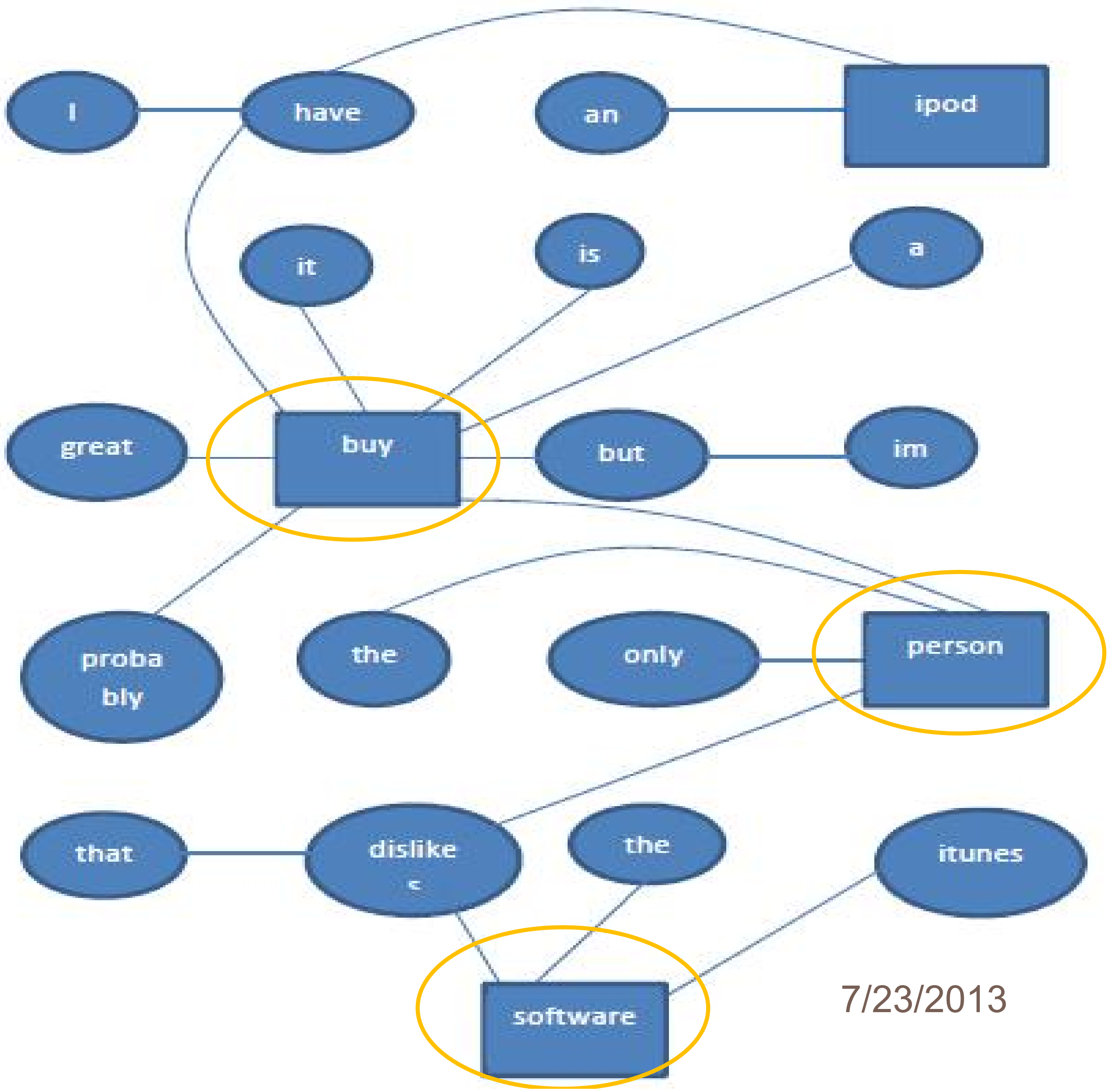
Clustering


I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

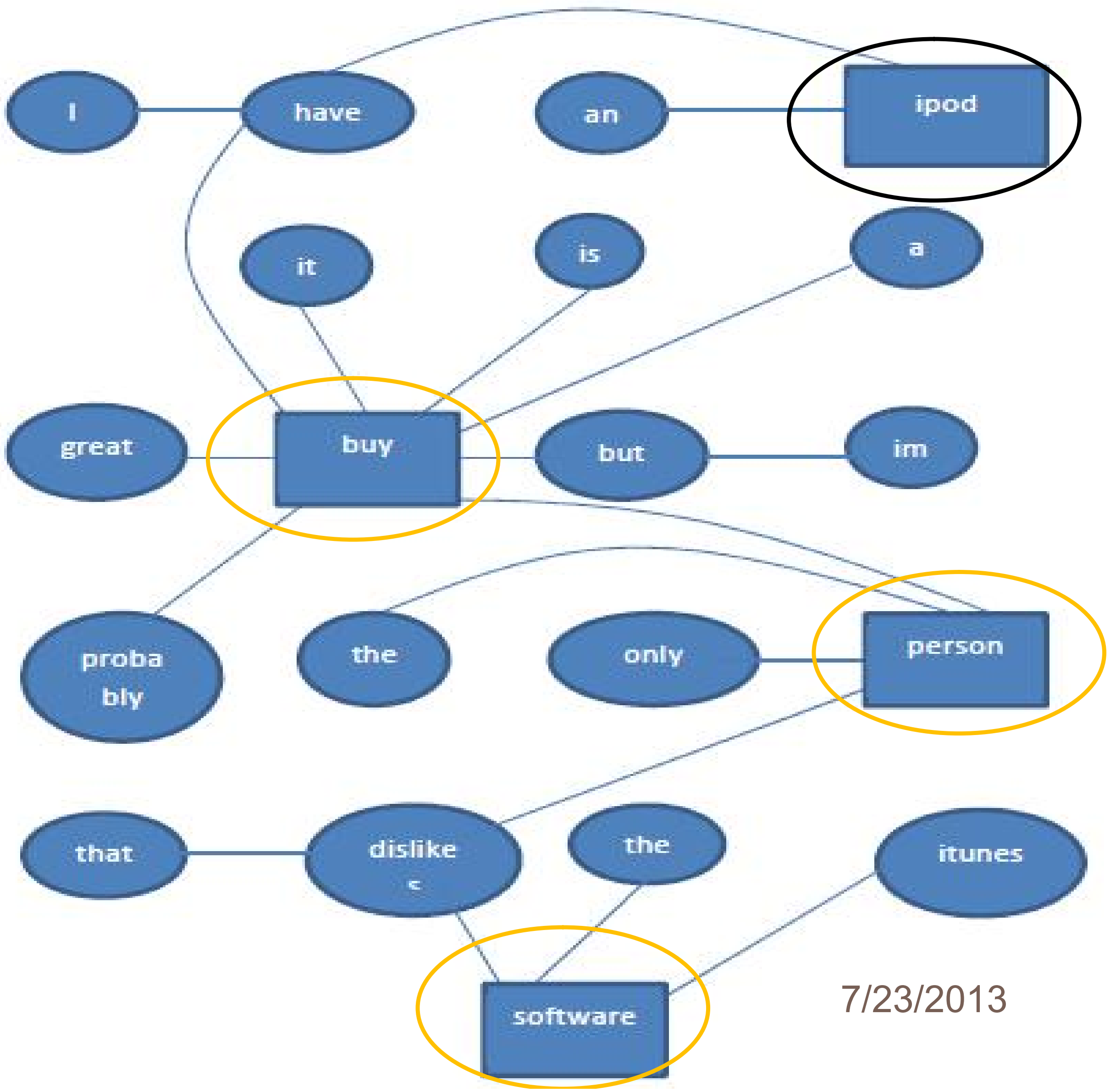
Clustering


I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

Clustering

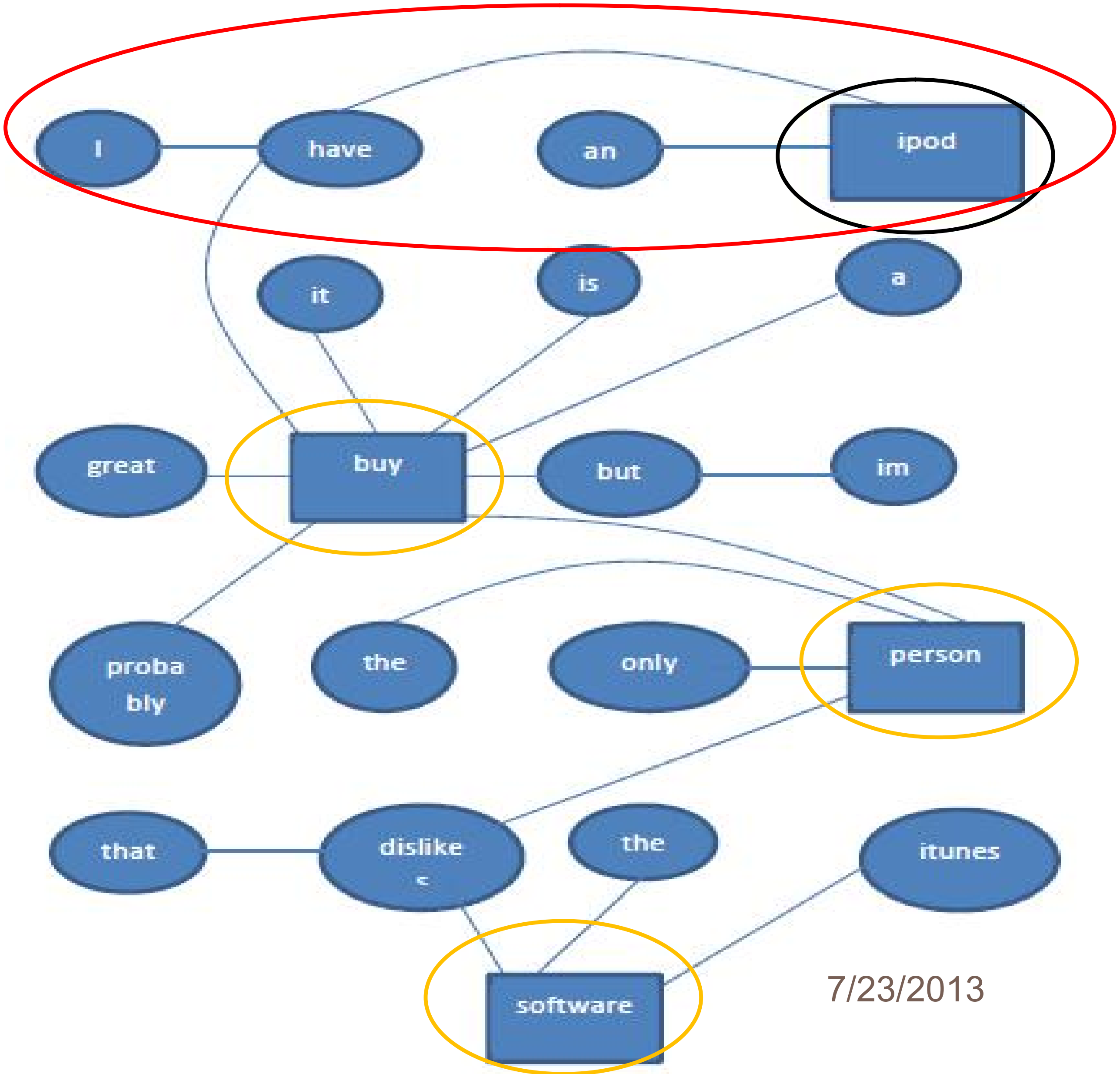

I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

Clustering

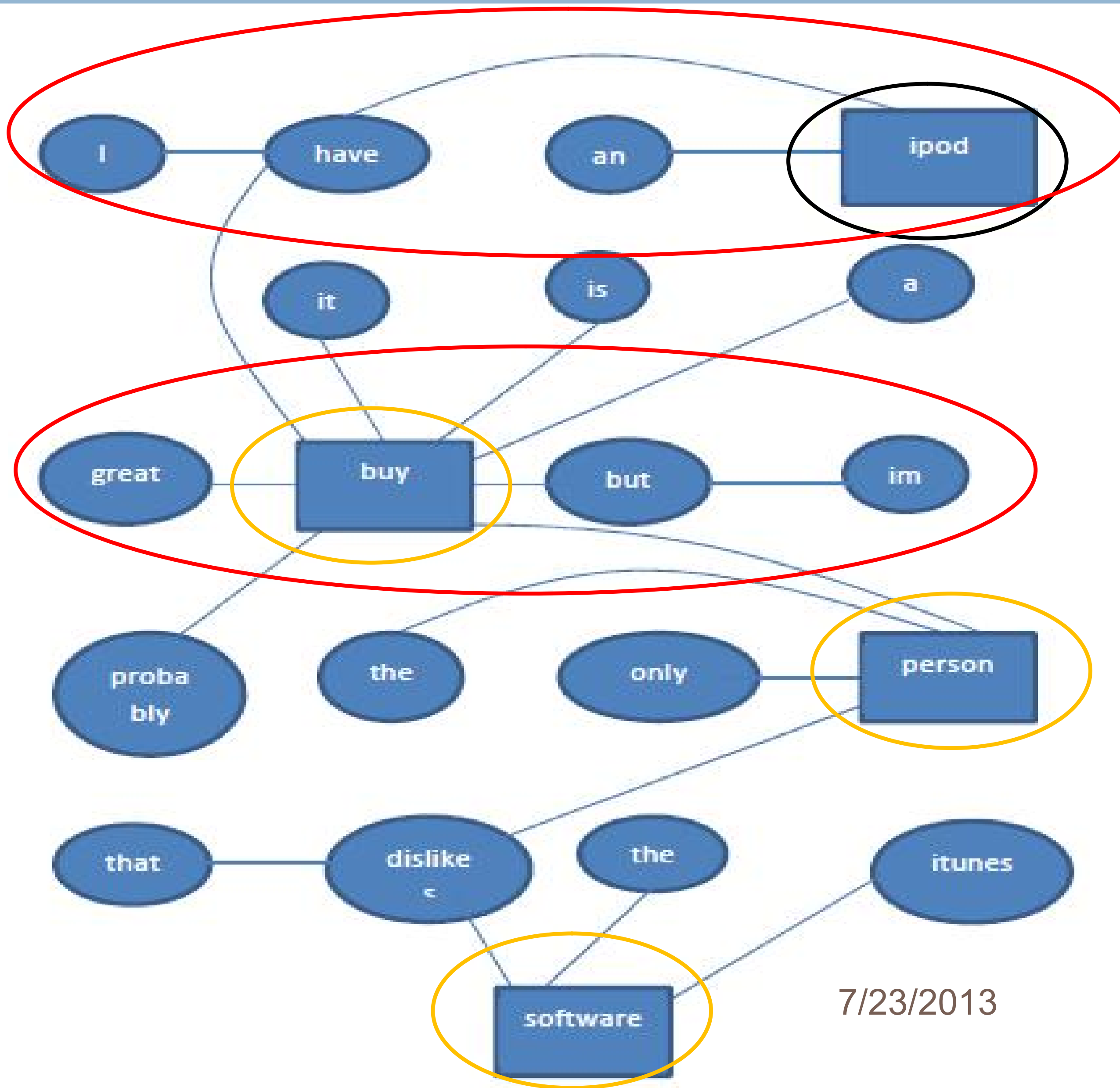


I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
c
the
itunes
software

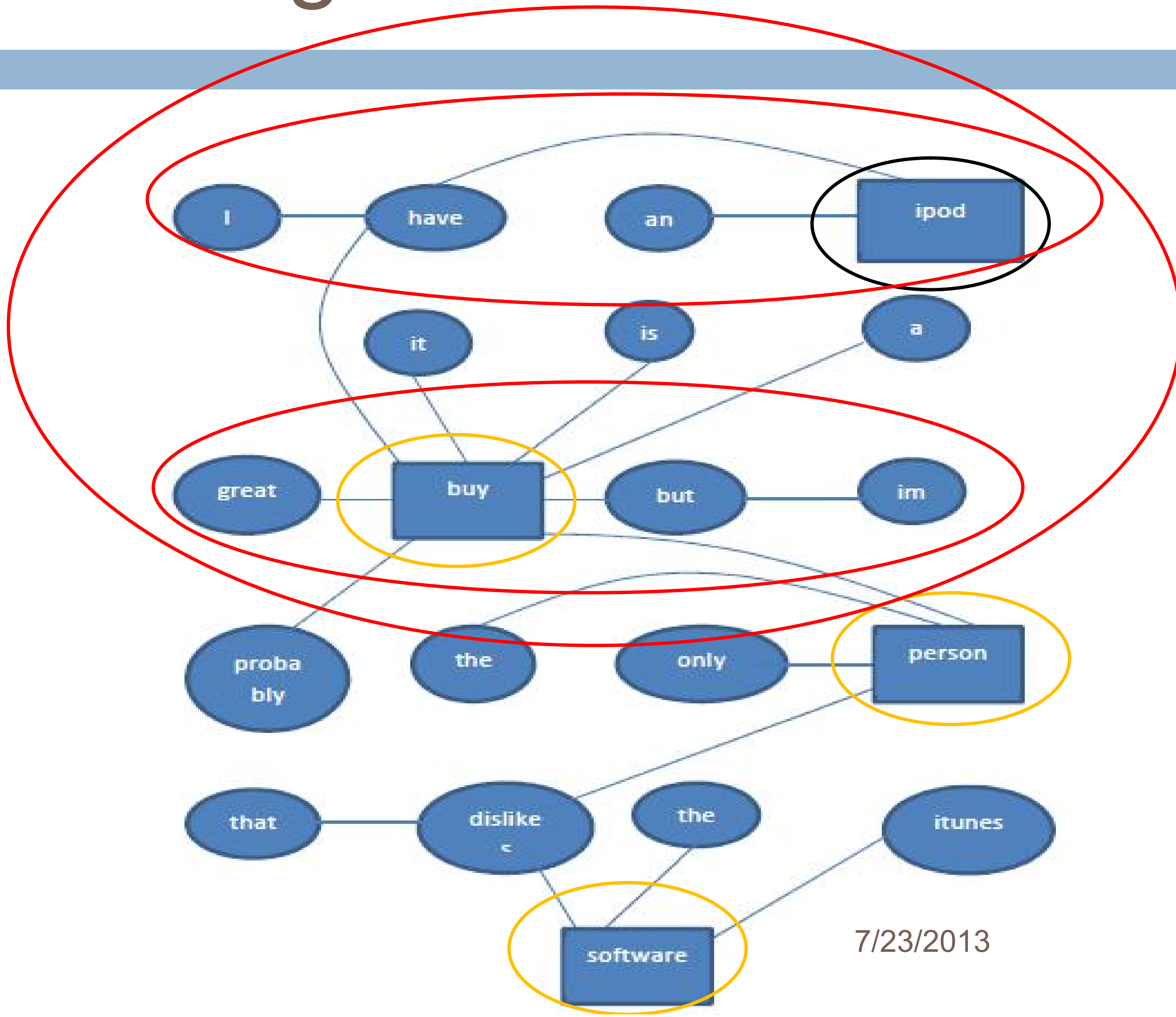
Clustering


I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

Clustering


I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

Clustering


I
have
an
ipod
it
is
a
great
buy
but
im
proba
bly
the
only
person
that
dislike
s
the
itunes
software

Pragmatics

Pragmatics

- *Elongation of a word, repeating alphabets multiple times* - Example: *happppyyyyyy*, *goooooood*. More weightage is given by repeating them twice

Pragmatics

- *Elongation of a word, repeating alphabets multiple times* - Example: *happppyyyyyy*, *goooooood*. More weightage is given by repeating them twice

- *Use of Hashtags* - *#overrated, #worthawatch.* More weightage is given by repeating them thrice

Pragmatics

- *Elongation of a word, repeating alphabets multiple times* - Example: *happppyyyyyy*, *goooooood*. More weightage is given by repeating them twice
- *Use of Hashtags* - *#overrated, #worthawatch.* More weightage is given by repeating them thrice
- *Use of Emoticons* - ☺ *(happy)*, ☹ *(sad)*

Pragmatics

- *Elongation of a word, repeating alphabets multiple times* - Example: *happppyyyyyy*, *goooooood*. More weightage is given by repeating them twice
- *Use of Hashtags* - *#overrated, #worthawatch.* More weightage is given by repeating them thrice
- *Use of Emoticons* - ☺ *(happy)*, ☹ *(sad)*
- *Use of Capitalization* - where words are written in capital letters to express intensity of user sentiments
  - *Full Caps* - Example: *I HATED that movie.* More weightage is given by repeating them thrice
  - *Partial Caps*- Example: *She is a Loving mom*. More weightage is given by repeating them twice

Spam Filter Evaluation

2-Class Classification

Tweets	Total Tweets	Correctly Classified	Misclassified	Precision (%)	Recall (%)
All	7007	3815	3192	54.45	55.24
Only spam	**1993**	**1838**	**155**	**92.22**	**92.22**
Only non-spam	5014	2259	2755	45.05	-

4-Class Classification

Tweets	Total Tweets	Correctly Classified	Misclassified	Precision (%)	Recall (%)
All	7007	5010	1997	**71.50**	**54.29**
Only spam	1993	1604	389	80.48	80.48
Only non-spam	5014	4227	787	84.30	-

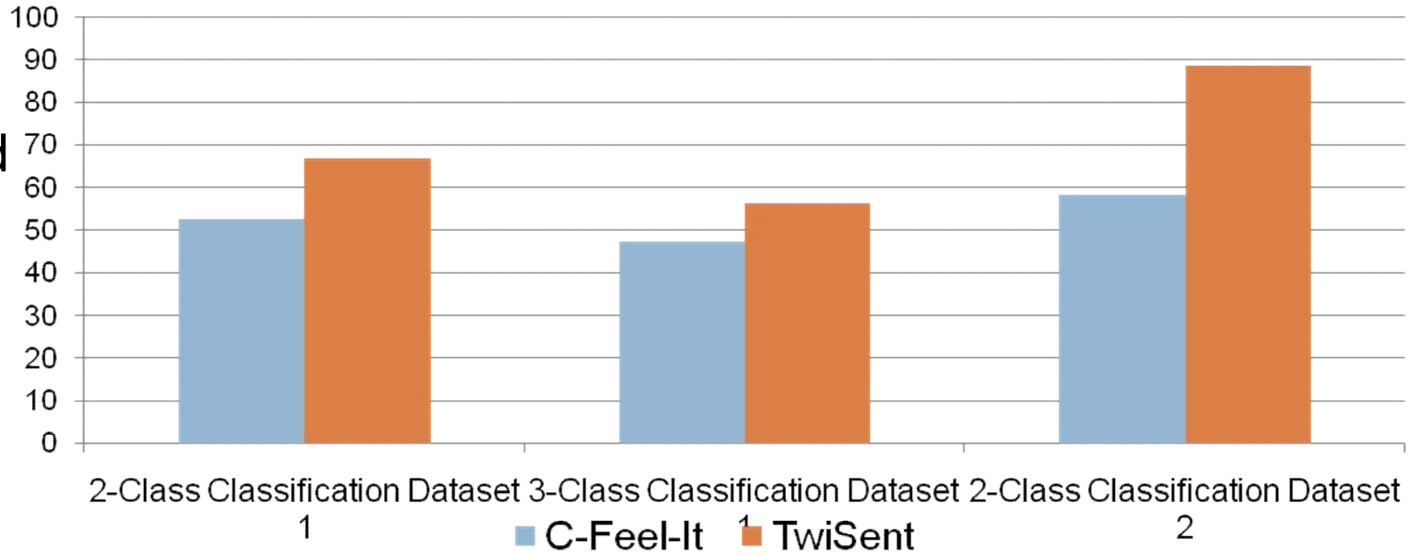
TwiSent Evaluation

TwiSent Evaluation

Lexicon-based
Classification

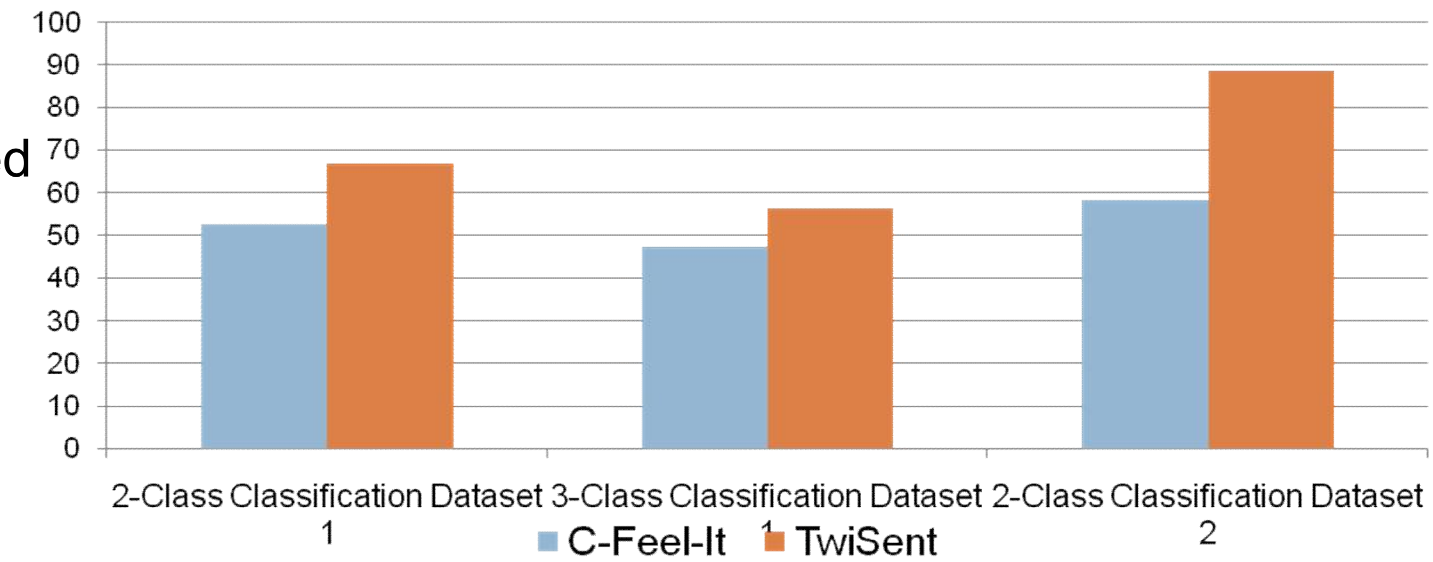
TwiSent Evaluation

Lexicon-based Classification



TwiSent Evaluation

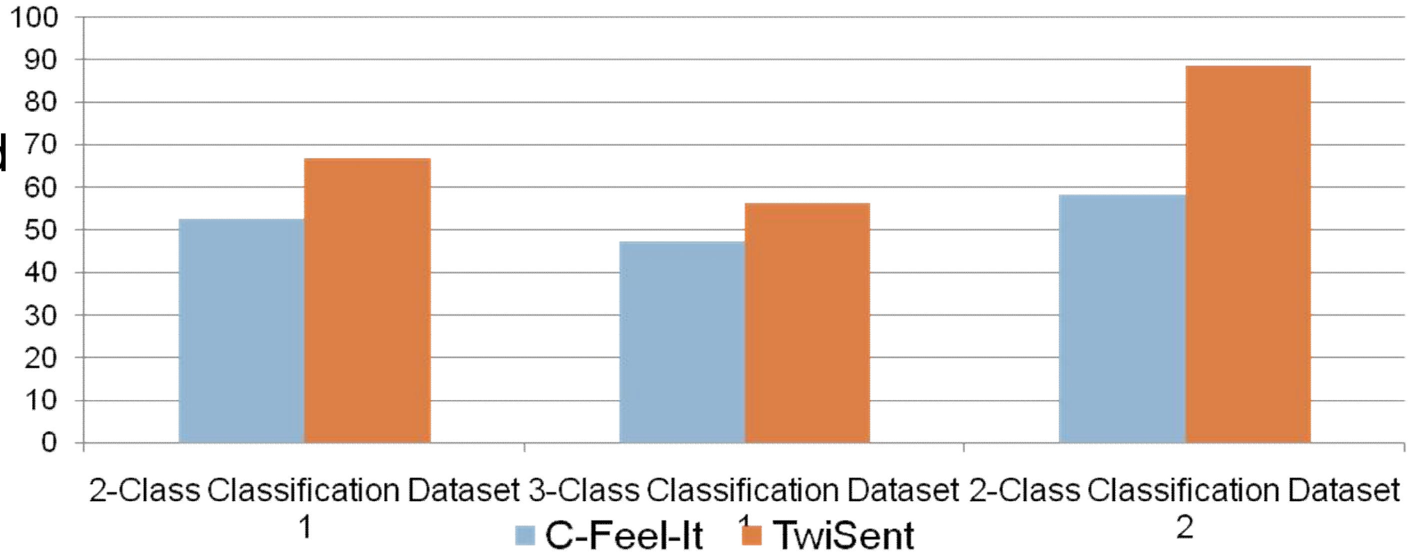
Lexicon-based Classification



Supervised Classification

TwiSent Evaluation

Lexicon-based Classification

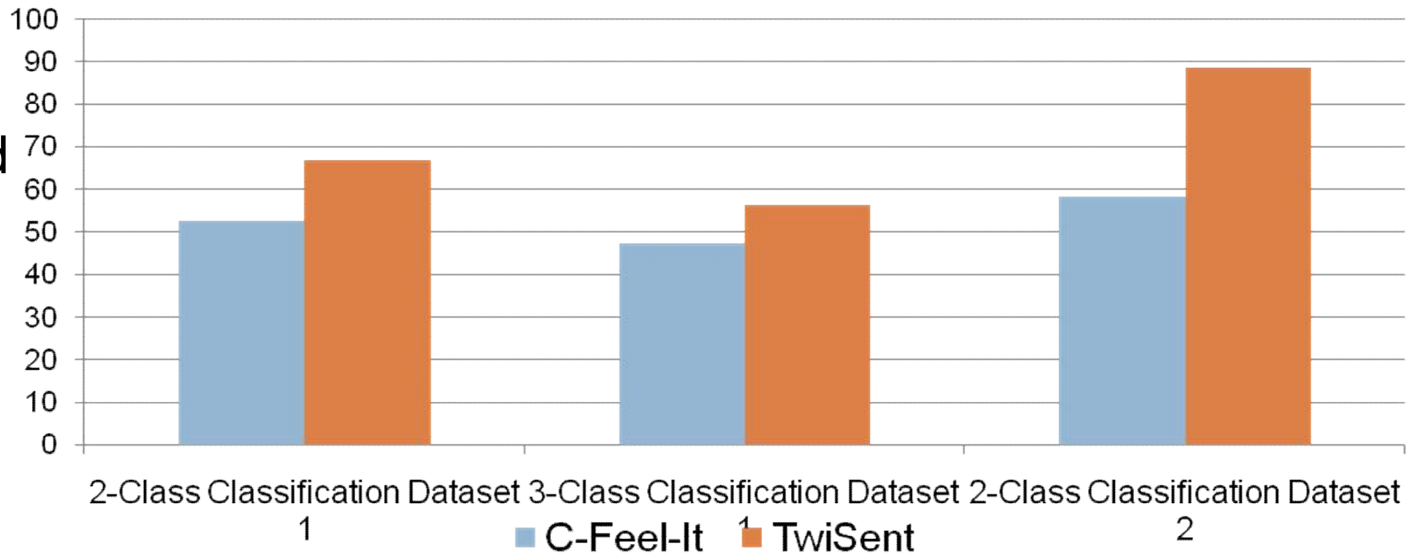


Supervised Classification

System	2-class Accuracy	Precision/Recall
C-Feel-It	50.8	53.16/72.96
TwiSent	**68.19**	**64.92/69.37**

TwiSent Evaluation

Lexicon-based Classification

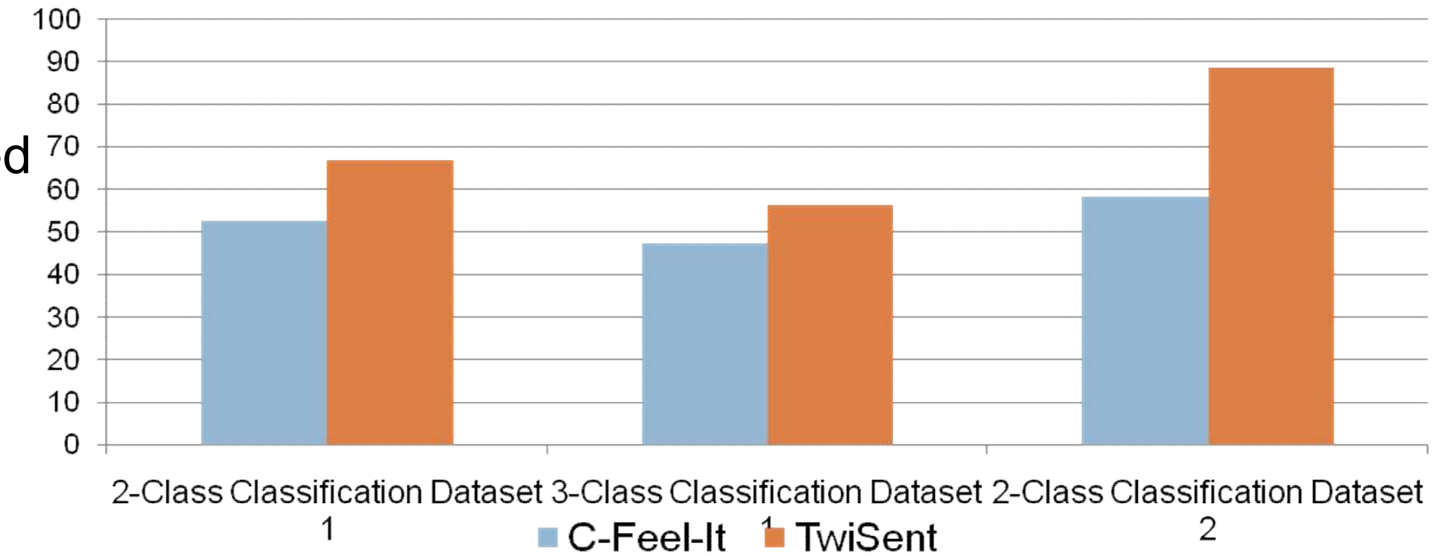


Supervised Classification

System	2-class Accuracy	Precision/Recall
C-Feel-It	50.8	53.16/72.96
TwiSent	**68.19**	**64.92/69.37**

TwiSent Evaluation

Lexicon-based Classification

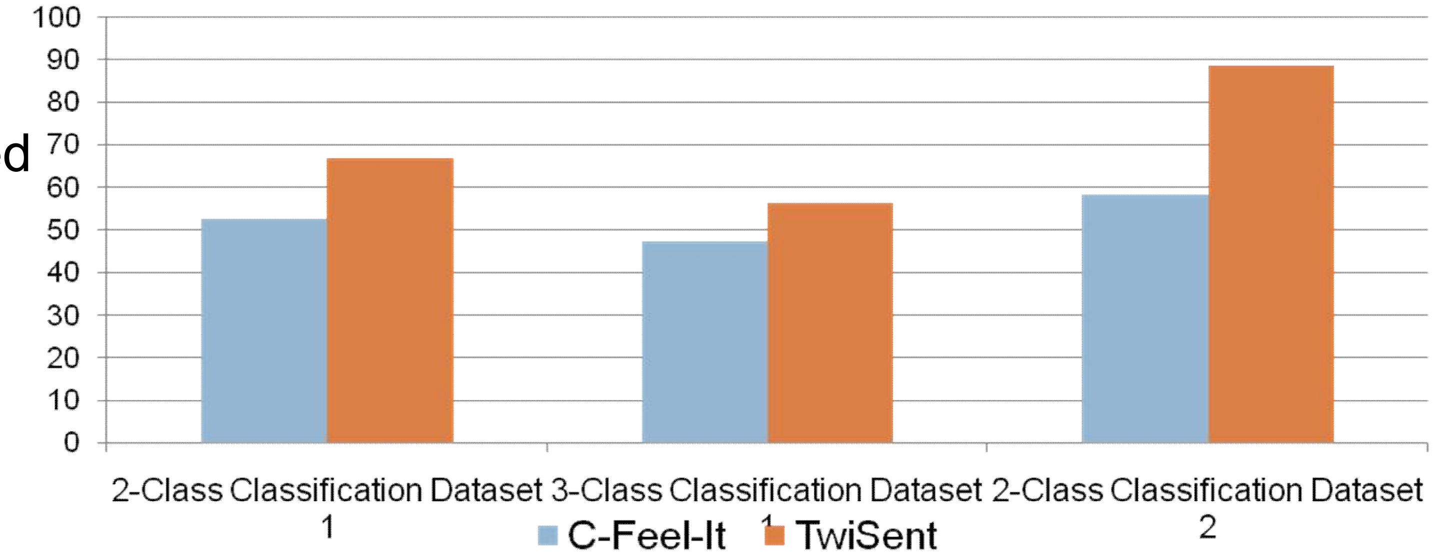


Ablation Test

System	2-class Accuracy	Precision/Recall
C-Feel-It	50.8	53.16/72.96
TwiSent	**68.19**	**64.92/69.37**

TwiSent Evaluation

Lexicon-based Classification



Ablation Test

Module Removed	Accuracy	Statistical Significance Confidence (%)
Entity-Specificity	65.14	95
Spell-Checker	64.2	99
Pragmatics Handler	63.51	99
Complete System	**66.69**	**-**