Identifying Demonstrative Lovers in Tango Texting App

Nicolas Poulallion
Stanford University
nicolasp@stanford.edu

Jean-Yves Stephan
Stanford University
jstephan@stanford.edu

***Abstract*—This paper aims at identifying the *demonstrative lovers* in Tango's chat interface, i.e. the users sending each other romantic *Tango Surprises*.**
Our initial approach was to use a *bag-of-words* representation of the conversations as well as additional user features that we defined in cooperation with Tango operational team. We then conducted a SVM classification with a Linear Kernel, reaching 80% accuracy but a bias vs. variance diagnosis revealed that there was potential in increasing the model complexity. A SVM with a RBF Kernel gave us indeed better results, but with unacceptable computation time.
Therefore, our second approach was to focus on an industrially operational solution, using only the conversations logs. The use of n-*grams* combined with optimized features normalization and selection enabled to build a program that could both accurately and efficiently identify *demonstrative lovers* based on their real-time conversations.

Keywords: Tango, text classification, supervised leaning, support vector machine, texting app.

Introduction

Tango is a social networking app enabling its 160+ million users to connect with friends by sending free text messages, making phone/video calls, and playing games. To personalize a text conversation, the users can send each other animated emoticons called *Tango Surprises* that pop up on the screen. They are widely used to help deliver your speech in a fun and illustrative way. Some of them are free, but there is also a large variety of premium Surprises that can be bought in thematic packages for $1.99 each.
The objective of this project is to predict whether two people are likely to send each other romantic *Tango Surprises* based on their recent conversation and additional user features. Eric Setton, co-founder and CTO of Tango, explained us there would be great value in targeting those specific users in marketing campaigns for Premium Surprises packages. It could also be a first step toward real-time suggestion of content-related *Tango Surprises*. Through weekly exchanges with Tango Data Analysis team, our project aims at providing a concrete, industrial-oriented solution.
Throughout this paper, we will use the notation *love* to refer to the parameters related to the conversations containing the selected romantic *Tango Surprises*. The notation *casual* will refer to the other conversations.

I. Experimental Design

In this section, we present the datasets used for our experiments and the features we constructed.

A. Datasets

We received 4 different .txt files from Tango:

- A file consisting of more than 214,000 *casual* conversations that took place in the US during the first week of November, with encrypted user IDs.
- Another file consisting of more than 96,000 *love* conversations, with the same geographic and temporal attributes.
- A file with users statistics, such as the gender and the number of messages sent as well as the number of *Tango Surprises* sent each day.
- A file with, for each conversation, the number of messages and *Tango Surprises* sent each day.

B. Features extraction

1) Conversations selection: First, in accordance with Tango, we decided to keep only the significant conversations, i.e. conversations with at least 3 messages and 15 words in total in our datasets. This first filter reduced our datasets to about 50,000 and 57,000 conversations respectively. In average, there are 33 messages per conversation and 6 words per message.

2) Conversations logs:

a) Words selection: Using *C++*, we used a stemming library (see [5]) to extract the root of the words. We introduced five additional tokens to capture non-verbal information:

- *sadsmiley* corresponds to sad emoticons, such as :(
- *happysmiley* corresponds to happy emoticons, such as :)
- *heart* corresponds to <3
- *brokenheart* corresponds to </3
- *url* replaces all the words containing *http* or *www.*

We decided to focus only on the 2,000 most frequent words in the entire corpus of conversations - after stemming. This decision does not affect our accuracy, as established in [3], but considerably improves our efficiency. Each conversation will then be represented as a vector of size 2000, containing the number of occurrence for each word in our dictionary.
For reference, we implemented a quick Naive Bayes to get an informal guess of the most relevant words, even if the Naive Bayes assumption is clearly not verified (for example, two different Spanish words cannot be considered as conditionally

independent). The 10 words with highest relative frequency ratio were: *amor*, *quiero*, *love*, *baby*, *mi*, *mucho*, *babe*, *por*, *kiss* and *te*.

b) Language: A striking observation was that the Spanish words seemed to be really significant, even those lacking a romantic overtone. This led us to add the language to our features. We created an additional boolean feature testing if the conversation is in Spanish. The percentage of Spanish conversations varies between 17% (*casual* conversations) and 28% (*love* conversations).

c) Number of words: For each conversation, we counted the total number of words.

d) Timestamp: For each conversation, we created a vector of size 24 containing the hourly number of messages sent. Fig. 1 shows the frequency for each hour for *love* and *casual* conversations.

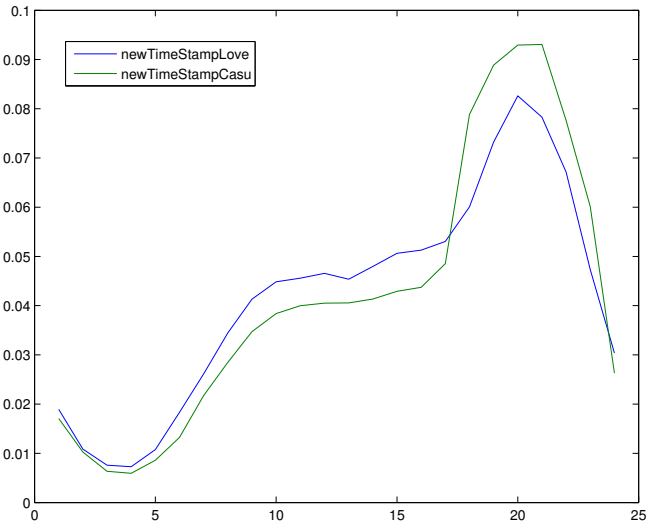

Fig. 1: Timestamps Frequency

3) Users Statistics:

a) Gender: For a given conversation, we indicated the gender of the two users. The data gives us 3 different genders: unknown, gender 1 and gender 2. According to [1], we decided to convert those categorical attributes into numeric data, i.e. 3 dimensional vectors defined as follows:

- $(1, 0, 0)$ for unknown.
- $(0, 1, 0)$ for gender 1.
- $(0, 0, 1)$ for gender 2.

b) Conversation Statistics: We indicated the number of messages and *Tango Surprises* sent each day, as well as the number of active days for each conversation.

c) Users Activity: For the two users involved in a given conversation, we stored the number of messages and *Tango Surprises* sent each day, as well as the number of active days on Tango.

Our final dataset therefore consists of 107,000 examples and 2,041 features.

II. Support Vector Machine - All Features

We conducted a SVM classification on all the Features.
In Section A we use a Linear Kernel since Linear Kernel SVMs are recommended when both numbers of instances and features are large as in our dataset (see [1]). We used the LIBLINEAR SVM library (see [2]) to implement a $L2$-regularized support vector classifier.
In Section B we implement a Gaussian Kernel using the LIBSVM library (see [1]) in order to reduce the bias.

A. Linear Kernel

1) Implementation and Parameter Optimization: To ensure the efficiency of our algorithm while keeping a good accuracy, we used the $L2$-loss primal solver.
We also chose to normalize our features to a $[0, 1]$ scale by dividing each feature by its maximum. Compared to other normalization methods, this scaling has the advantage of preserving the sparse property of the design matrix, which has a major impact on the algorithm efficiency (normalizing by mean and variance multiplies computation time by ~ 5 while giving similar results).
In order to set a good trade-off between efficiency and computation time, we tested the 10-fold cross validation accuracy on exponentially increasing values of the *cost parameter* C. The curve on Fig. 2 ends with an asymptotic value, as indicated in literature (see theorem 3 in [4]), proving that $C = 2^5$ is a fair choice.

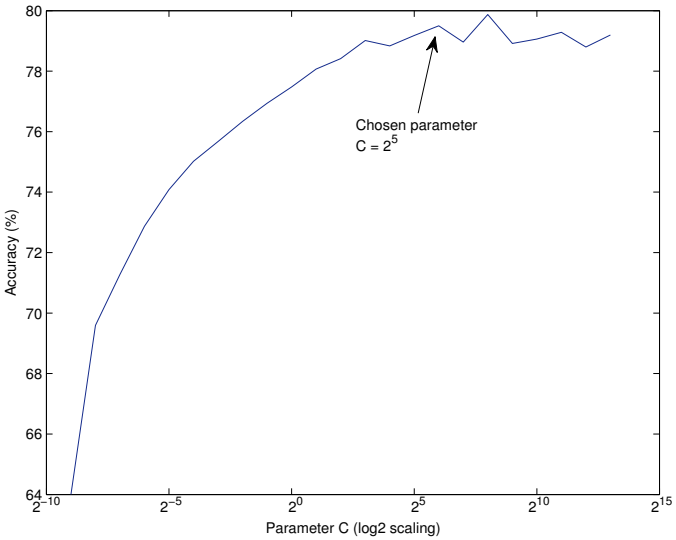

Fig. 2: Cost Parameter Optimization

2) Results and Performance: We split the dataset randomly into a training dataset (75,000 examples corresponding to 70% of the total dataset) and a test dataset (32,000 examples or 30% of the total dataset) and then computed the classification - in $\sim 10s$ with Matlab. Classification outputs on the train dataset are shown on table I.

The accuracy rate reaches 80%, improving significantly our initial quick Naive Bayes approach ($\sim 58\%$). The recall

		Predicted	
		casual	*love*
Actual	*casual*	10,882	3,596
	love	2,674	14,197

TABLE I: Classification outputs (Accuracy: 80.0%)

rate, measuring the frequency of *love* conversations that our algorithm successfully identifies, is of 84%. This rate is our principal metric, since Tango does not want to miss any potential *love Tango Surprises* customer. The precision rate - 80% - is also an important metric because Tango wants to avoid flooding its users with marketing offers that do not concern them.
To make a Bias vs. Variance diagnosis, we repeat the same analysis on increasing (random) sample of our dataset. The learning curve we obtain is shown on Fig. 3.

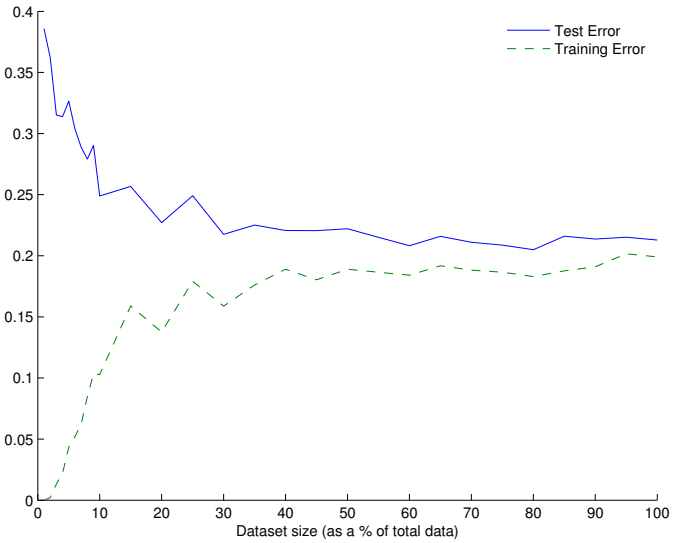

Fig. 3: Learning Curve

The test error stabilizes at about 20%: our dataset is large enough to train the algorithm so gathering more data is not an issue. However, the gap between the training and test error reduces down to ∼1%, which is very small. It suggests that our model may underfit the data (Bias issue), and that we should increase the model complexity to try to get better results.

B. Radial Basis Function Kernel

Using a more complex kernel appears to be a good way to increase our model complexity, since it allows to grasp more interactions between our features. Keeping the same normalization as previously, we tested the 5-fold cross validation accuracy of our training set over a grid of model parameters (C, γ).

This analysis showed that the best couple (C, γ) was $(2^5, 2^{-8})$ with a cross-validation rate equal to 82.03%, as shown on Fig. 4.

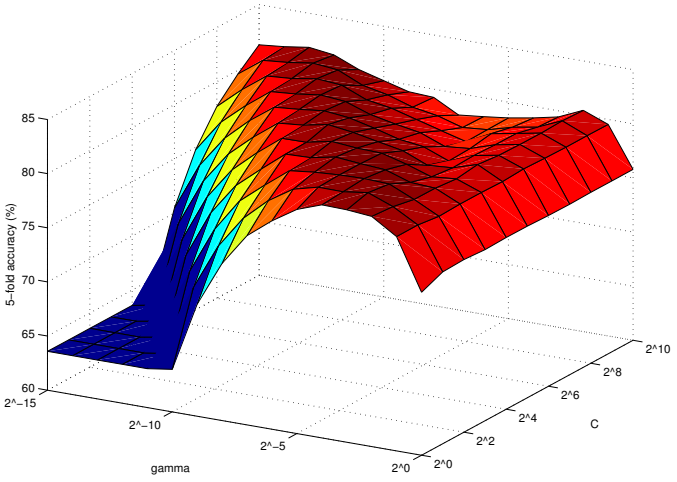

Fig. 4: C and γ Selection

Although this result was slightly better than the one we previously obtained with the Linear Kernel, the computational time (∼5 hours) cannot be used by Tango in an integrated industrial system. The next section will therefore focus on how to achieve an accurate and easy-to-compute classification.

III. Operational Solution - Conversations Logs

To create an industrially operational solution that Tango could easily integrate to its system, we will now only use the conversation logs, i.e. the conversation itself and the associated metadata. We will not use the *Users Statistics* anymore, as it requires accessing another database, which cannot be done in real time. Furthermore, the *Users Statistics* we used in the previous model probably introduced a bias. In particular, the number of *Tango Surprises* sent each day is correlated to the probability of sending a *love Tango Surprise*.
This choice will decrease a little bit our accuracy, but allows us to save computation time and thus makes it more suitable to a real-time application.

A. n-grams

We switched from the naive *bag-of-words* representation to a more complex one, using contiguous sequences of n words or n-*grams* (unigrams, bigrams and trigrams).
We fixed a threshold to focus only on the 100,000 most frequent trigrams (out of more than 3,500,000) and the 50,000 most frequent bigrams (out of more than 700,000) and kept our previous 2,000-unigram dictionary. Here again, setting this threshold notably increases our efficiency without really lowering our accuracy (cf. [3]).

B. Features Selection

Using such a large set of unigram, bigram and trigram features requires a good selection method so as to prevent overfitting issues. Since traditional methods such as forward

or backward search are computationally intractable, we chose to filter the top k_1 (respectively k_2, k_3) best unigrams (respectively bigrams, trigrams) according to a score function. We chose the Bi-Normal Separation metric, which substantially outperforms most other metrics as indicated in [3]. If tpr (respectively fpr) denotes the true (respectively false) positive rate i.e. the frequency of *love* (respectively *casual*) conversations containing a given n-*gram*, then the BNS score is simply $F^{-1}(tpr) - F^{-1}(fpr)$ where F^{-1} is the standard Normal's distribution inverse cumulative probability function. For intuition, this score measures the distance between two hypothetical thresholds, such that a random draw from the normal distribution exceeding this threshold would trigger the occurrence of the n-*gram* in a *love* (respectively *casual*) conversation, as shown on Fig. 5.

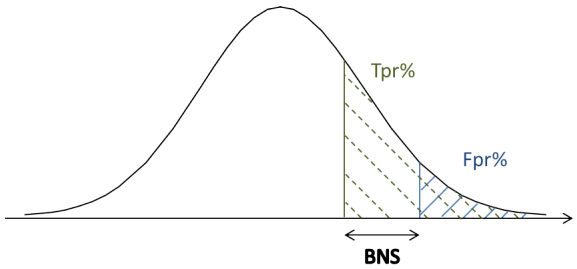

Fig. 5: Bi-Normal Separation Metric

Table II gives an example of the 15 most significant *love* n-*grams* according to the BNS metric. A high score is usually given to n-*grams* that are semantically and statistically significant (improving our classification accuracy).

Unigrams	*Bigrams*	*Trigrams*
beso	te amo	te amo mi
amo	mi amor	mi amor te
amor	love you	the amo y
kiss	si amor	i love you
lindo	ok amor	i love u
corazón	te quiero	te mando un
bello	a kiss	babi i miss
love	amor no	love you too
tokenheart	love u	good morn my
quiero	un beso	love u so
babe	mi corazón	tenga un lindo
encanta	hola amor	babi i love
miss	miss you	call me i
babi	quiero mucho	ya me voy
hermoso	ok bb	i miss you

TABLE II: Most significant *love* n-*grams* using BNS

Having computed that score for each n-*grams* feature, we then used 5-fold cross-validation to determine the right k_1 (respectively k_2, k_3) number of unigrams (respectively bigrams, trigrams) to select. Optimizing these values independently (i.e. by doing the classification using only unigrams, bigrams or trigrams) led to choosing $k_1 = 2,000$, $k_2 = 4,000$ and $k_3 = 10,000$. The whole unigrams dictionary is therefore selected - this is no surprise since our analysis of Section II showed that the *bag-of-words* model underfitted the data. However, a saturation phenomenon appears when more and more bigrams (respectively trigrams) are used, as shown on Fig. 6. The reason being that adding more features above the optimal threshold leads to overfitting issues.

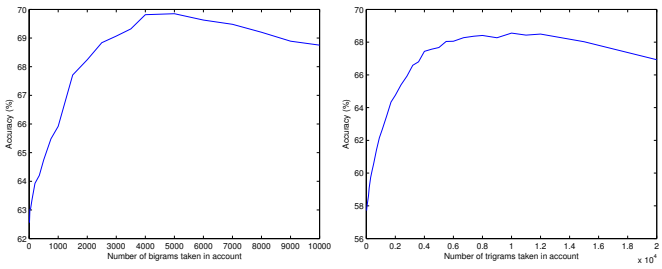

Fig. 6: 5-fold cross-validations led independently on bigrams and trigrams reveal the optimal number of features to select

C. Normalization

Before performing feature-wise normalization to a $[0, 1]$ scale as in Section II, we used the term frequency-inverse document frequency (TF-IDF) statistic for conversation-wise normalization. This metric takes the raw term frequency of a given n-*gram* in a given conversation, and scales it by a multiplicative factor that depends on the occurrence of the n-*gram* over *all* conversations. Precisely, for a n-*gram* w in a conversation $c \in C$, the TF-IDF weight is calculated by:

$$\text{TF-IDF}_{w,c} = f_{w,c} \times log \frac{|C|}{|\{c \in C : w \in c\}|}$$

The second term can be seen as a discriminatory power factor that emphasizes the rare words and diminishes the non-significant common words. The combination of these two normalization steps constitutes an important improvement in our approach, increasing accuracy by at least 4%. It also ensures an efficient computation on the LIBLINEAR library, while preserving the sparse property of the matrices, which is absolutely necessary considering the scale of our problem.

D. Results

Over our complete 110,000 examples dataset, using the previously described normalization and features selection method, we searched the optimal values of the number of unigram, bigram and trigram features to select by computing 5-fold cross validation over three nested loops. It resulted in keeping the entire 2,000-unigram dictionary and adding the top 2,000 bigrams and 4,000 trigrams according to the BNS score. With this choice of parameters our classification algorithm reaches an accuracy of 82.1% on the test dataset (30% of total data, see table III), which outperforms the Section II approach even though the informative user features were dropped off. Computation time is about one minute on a MATLABx64 on the Corn cluster.

		Predicted	
		casual	*love*
Actual	*casual*	14,588	3,336
	love	2,658	12,848

TABLE III: Classification outputs (Accuracy: 82.7%)

The feature selection approach has the nice advantage of setting a good trade-off between bias and variance issues, whatever the dataset size: if more examples were given, more bigrams and trigrams would be used in the classification. Fig. 7 shows the training and test error of the classification with these fixed parameters, on an increasing random portion of the dataset size (average over 5 runs). Compared to Fig. 3, the test error now only stabilizes when the whole dataset is used, which shows an intermediate stage between a definite bias issue and a definite variance issue.

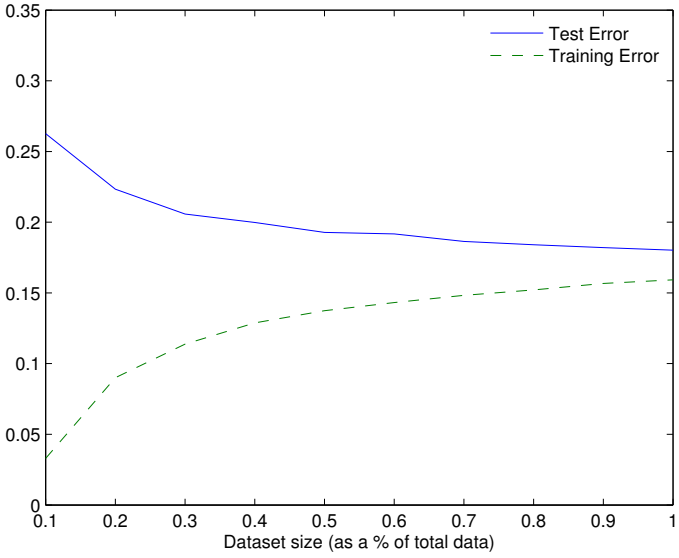

Fig. 7: Final Learning Curve (average over 5 runs)

Finally, let us give an overview of the approaches of increasing complexity that we developed in this paper (see Fig. 8). The initial Naive Bayes classification gave but little accuracy improvement compared to the 50% reference (random guessing taking into account the frequency of *love* and *casual* conversations). The switch to a SVM approach, based on words count and on conversation logs (*metadata*) has the most significant impact. Adding bigrams and then trigrams mitigates the underfitting issue that we diagnosed in Section II, allowing a significant 6% increase in accuracy. But the tendency shows that adding 4-grams would probably have little impact for a very high computation cost. Tango was pleased with the 82% final accuracy, given the biased nature of the dataset (all lovers do not send each other romantic *Tango Surprises*)!

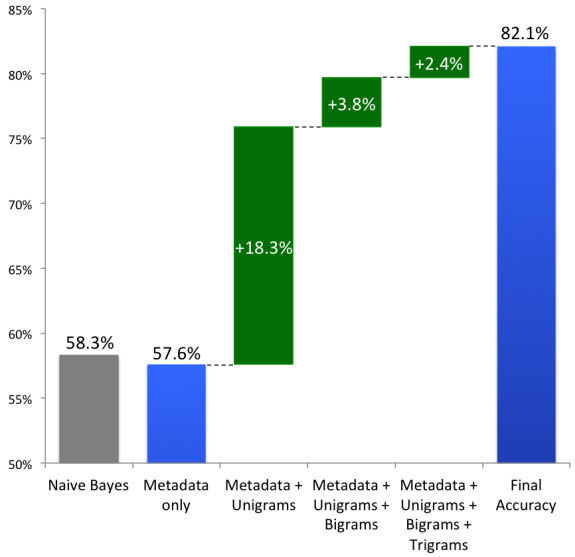

Fig. 8: Accuracy improvement through increasing model complexity

Conclusion

In this project we implemented a supervised binary text classifier. Using a very large dataset of real conversations between Tango users, we were able to predict which users are likely to send each other romantic *Tango Surprises*, with an 82% accuracy. We particularly focused on building an operational solution, which can process real-time conversations logs and identify the *demonstrative lovers* in Tango's chat interface. Our progression led us to tackle many of the main topics and practical techniques in the field of text classification: Support Vector Machines - where the Linear Kernel combined accuracy and efficiency- as well as features normalization (TF-IDF metric) and selection (Bi-Normal-Separation scoring). We moved from an initial *bag-of-words* approach which suffered from bias issues, to a combination of n-*grams* ($n \leq 3$), which solved the under fitting issue by adding the right amount of significant information to our model.

Acknowledgment

We would like to thank Eric Setton, co-founder and CTO of Tango, and Guillaume Sabran, data analyst at Tango for their precious help and interest in our project. We are also thankful to Andrew Ng and the CS 229 Teaching Staff for this great class.

References

[1] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, *A Practical Guide to Support Vector Classification*, 2010.

[2] Rong-En Fan, Kai-Wi Chang, Chih-Jen Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, LIBLINEAR*: A Library for Large Linear Classification*, 2012.

[3] George Foreman, *An Extensive Empirical Study of Feature Selection Metrics for Text Classification*, 2003.

[4] S. Sathiya Keerthi, and Chih-Jen Lin, *Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel*, 2003.

[5] *Oleander Stemming Library*, Copyright 2012, Oleander Software, Ltd. All rights reserved, (available at http://www.oleandersolutions.com/stemming/stemopensource.html).