Smart Chat

Rishi Dua
http://github.com/rishirdua

Harvineet Singh
https://github.com/harvineet

Real-time chat-based personalized recommendation system

1 Smart Chat

Smart Chat is a chat based application which gives the user live suggestions based on his/her chat content. For example, when the user has a discussion about movies in his chat the application will look up movie timings and ratings and he will be recommended other movies based on his interests. Similarly the user will be suggested hotel and train availability, weather and directions information etc. based on the contents of his chat. The chat is categorized into certain pre-defined category with each category having specific actions to integrate search, schedule management and chat.
NLP techniques are used to extract user interests and detect the category of the chat for the recommendation system. With time, the algorithm learns the user preferences better based on users response to the recommendations giving the user a more personalized experience.
Smart chat implements and evaluates several algorithms in the context of developing a recommender system based on data gathered from Facebook chat. The algorithms draw from principles and techniques in Machine Learning, Natural Language Processing, Information Retrieval, as well as Graph Theory.

2 Motivation

Chat apps are generally available in all social networking platforms. Apart from emoticons and file sharing, they are plain peer to peer interactions and with very limited additional features. We try to integrate personalized recommendation and user profile and interest analysis system similar to the ones implemented by search engines and e-commerce websites within a chat application. This gives the user a convenient way to have personalized recommendations given to him/her live during his/her conversations, rather than have him/her browse for them otherwise. This also includes maintaining an activity calendar for user reminding him of his schedule while planning tentative works on chat.
Personalized recommendation systems [1][2][3] are implemented by Google, Yahoo and Bing for search. Similarly Amazon, eBay and other e-commerce websites have personalized recommendation systems for buyers. However as of now, no website or application provides real-time recommendation system for chat.[4][5] We are working on implementing the features of search and

shopping personalization into a real time recommendation system for chat software by focusing on reducing the computational time needed to generate the recommendations.

3 Installation

1. Install server
   (a) Install tasksel sudo apt-get install tasksel
   (b) Install LAMP Server from tasksel
   (c) Copy the files from src/smart-chat to where you want to host (eg: var/www/smart-chat or htdocs/smart-chat/)
2. Install scikit-learn from http://www.lfd.uci.edu/ gohlke/pythonlibs/
   (a) Install dependencies: sudo apt-get install build-essential python-dev python-setuptools python-numpy python-scipy libatlas-dev libatlas3gf-base
   (b) Download source
   (c) Extract source and build with sudo python setup.py install (Make sure you have c++ installed) note: Don't use 3rd party distro as it gives an error
3. Install beautiful soup (http://www.crummy.com/software/BeautifulSoup/)
   (a) sudo apt-get install python-bs4
4. Download nltk and nltk data (NOTE: Use sudo -E if you need user variables eg: proxy settings)
   (a) Install settuptools (http://pypi.python.org/pypi/setuptools) wget https://bootstrap.pypa.io/ez_setup.py sudo pyhon ez_setup.py
   (b) Install Pip: run sudo easy_install pip
   (c) Install NLTK: run sudo pip install -U nltk
   (d) Install wordnet data. Run python and type import nltk nltk.download()

   Ensure that server is running using root user account and the nltk.data directory is downloaded for that user account

5. Configuration

   (a) config.py (proxy settings for server's internet)

   (b) config.php (most options are self-explanatory)

4 Implementation

We have implemented text classification (pre-processing, feature extraction, feature selection, classification) for general text which can be categorized into predefined types: movies, products, food, hotels, cars etc.
We bootstrapped our algorithms with the following data:

- Movie: Polarity v2.0, Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs up Sentiment Classification using Machine Learning Techniques, Proceedings of EMNLP 2002.
- Products: Amazon Product Review Data (more than 5.8 million reviews) used in (Jindal and Liu, WWW-2007, WSDM-2008; Lim et al, CIKM-2010; Jindal, Liu and Lim, CIKM-2010; Mukherjee et al. WWW-2011; Mukherjee, Liu and Glance, WWW-2012)
- Restaurant: Yelp reviews, Phoenix Academic Dataset
- Cars: UCI ML repository, OpinRankDataset 2008
- Hotels: New Delhi Hotel reviews, UCI ML repository

The current algorithm implements the following steps on the above dataset for training:

1. Pre-processing: Stop words removal, Stemming using WorldNet lemmatizer
2. Feature extraction: TF-IDF of word count vectors
3. Feature selection: Select k-best features using Chi-square feature selection
4. Classification: Multinomial NB, Bernoulli NB, LinearSVC, Perceptron. Cross-validation on the current dataset has f1- score in decreasing order for LinearSVC algorithm, Multinomial, Bernoulli NB, Perceptron (implemented using scikit-learn library)

5. Generation of recommendation based on detected topic (label)

We implement the algorithm as an Ajax based PHP chat application. The following screenshot shows the User Interface developed using HTML5, jQuery JavaScript library [6] and Bootstrap framework [7].

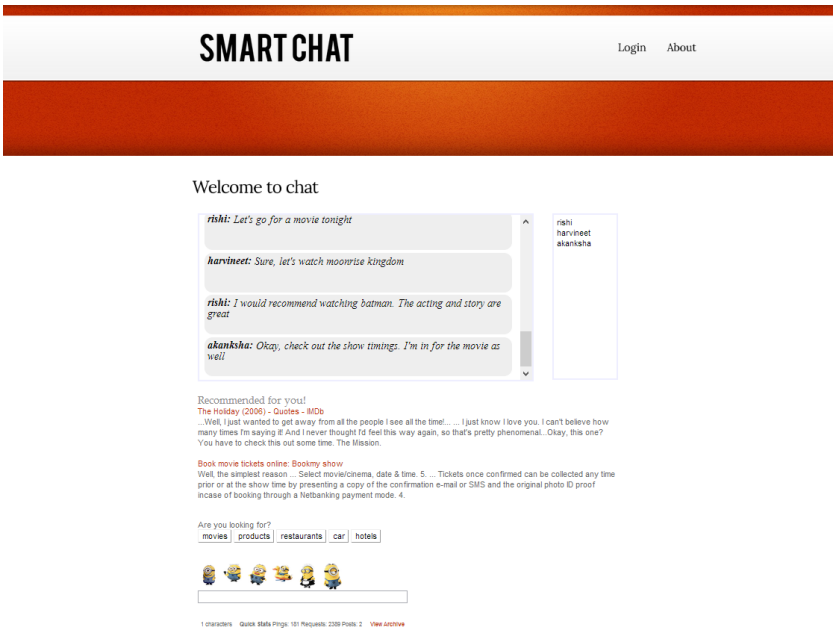

Figure 1: Screenshot of the User Interface

We have implemented the following features in the application:

- Extending to larger training dataset (from chats through the system) and implementing the algorithm for more categories
- Finding correct time and context to provide recommendation to the user so as to minimize hindrance to the chat. We aggregate and classify recent sentences to better identify context, sentences with interrogative questions to recognize need for suggestion and also to identify non-topical discussion so that no recommendation is given. If last 3 sentences of a users chat are from the same topic, only then a recommendation is shown

- Ranking of the suggestions based on user's past actions and the context. Training of user preferences prediction based on his response to the generated recommendation
- Detecting category and then searching related pages to show relevant suggestions rather than just Google search. e.g.: if someone says cheap car, suggest Tata Nano, or someone says tallest building, suggest Burj Khalifa.
- The training corpus gets expanded every time a sentence classification is validated by the user by clicking the link suggested. This makes the algorithm to work better on chat rather than structured text.
- Taking feedback from user to improve classification. If a sentence is incorrectly classified and a user reports an error, the correction is incorporated in the model.

5 Results

For testing the improvement by using stemming (lemmatization), we get the following results using all words as feature vectors

	Not Lemmatized	Lemmatized
Multinomial NB	0.9694(0.0047)	0.9706(0.0044)
Bernoulli NB	0.9681(0.0041)	0.9685(0.0043)
Linear SVM	0.9701(0.0038)	0.9702(0.0055)

Table 1: Accuracies and standard deviation (in brackets) for different pre-processing algorithms

Thus, lemmatization gives us the better cross validation accuracies and we include it in the pre-processing step for the application. For selecting the best classifier, we try out Bernoulli Nave Bayes, Multinomial NB, Linear SVM (L1 and L2), Perceptron and Ridge classifier. [8] The graph of obtained learning accuracies is shown below
Since we are using the application for real-time recommendations, we also calculate and plot the relative training and prediction time for the same dataset. Prediction time is a crucial factor for our model.

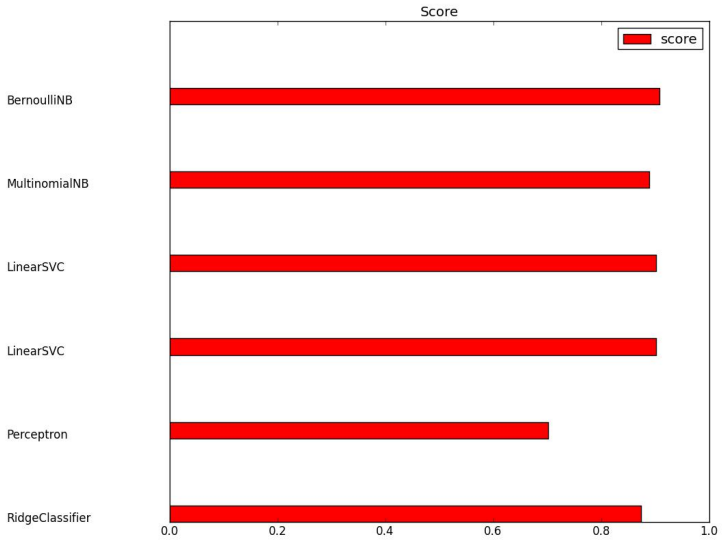

Figure 2: Testing accuracy of different classifiers

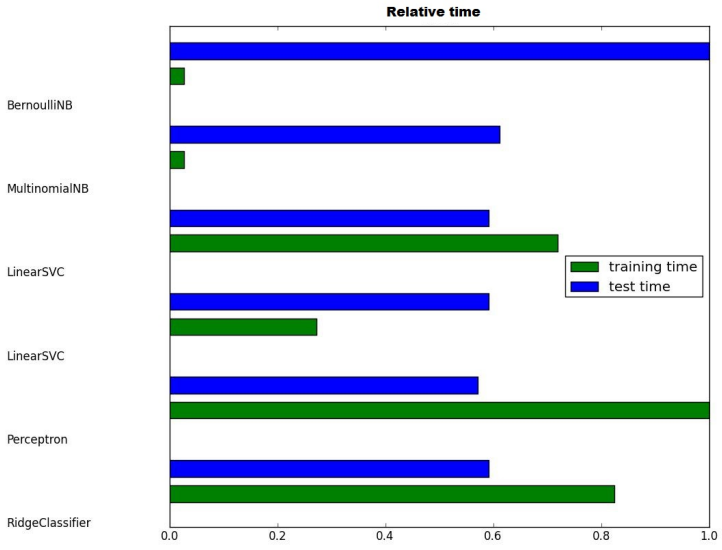

Figure 3: Relative training and testing times for different classifiers

Based on the above results, we observe linear SVM and Multinomial have almost similar accuracies. We choose Multinomial as the preferred model due to its lower training time.

For selecting the number of features to use, we calculate the 10-fold cross validation accuracies for different number of features. For this, we choose the best k features using Chi-Square measure. On plotting the accuracies v/s the number of features, get the following results

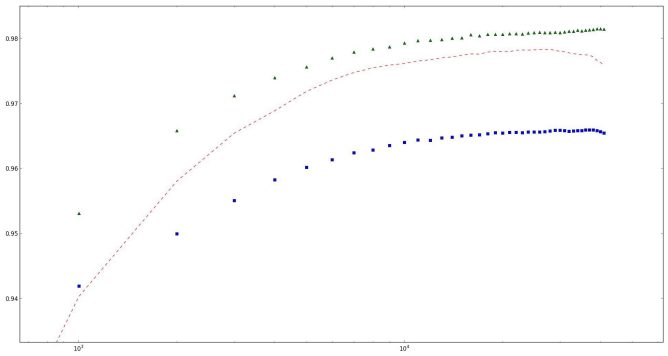

Figure 4: Accuracy (10 Fold Cross Validation) vs. K best features (Log scale)

Based on the results of the above step, we observe Multinomial outperforms Bernoulli NB [9] for large vocabulary and high accuracy is obtained by using considerably low number of features (around 5000 as seen from Figure 3). We observe that the curve begins to saturate after a certain number of features. Thus we conclude that there are redundant word features.

6 Future scope

Further development can be done in the following areas:

- Personalization of the feature extraction technique, clustering of users based on their preferences.
- Maintaining an activity calendar for user reminding him of his schedule while planning tentative works on chat.
- Changing the application from a Ajax based chat to an XMPP chat server [10]
- Implementing and testing multiclass and multilabel algorithms

- Integrating with Facebook chat API [11]

7 Source

Smart Chat is released under the terms of the MIT license.
The MIT License is simple and easy to understand and it places almost no restrictions on what you can do with a Smart Chat. You are free to use Smart Chat in any other project as long as the copyright header is left intact

8 Credits

The developers of smart-chat would like to thank Dr. Parag Singla, Indian Institute of Technology Delhi for guiding his throughout the development. We would like also like to thank Akanksha Jain and Kangkan Boro for helping us with the development.
This application uses Open Source components. You can find the source code of their open source projects along with license information below. We acknowledge and are grateful to these developers for their contributions to open source.

- **Chatr**
  Website: http://www.sterryit.com/chatr/
  Code: https://github.com/weex/Chatr
  Licence: BSD Licence

- **scikit-learn**
  Website: http://scikit-learn.org/
  Code: https://github.com/scikit-learn/scikit-learn
  Licence: New BSD License

- **Beautiful Soup**
  Website: http://www.crummy.com/software/BeautifulSoup/
  Code: https://code.launchpad.net/beautifulsoup
  Licence: MIT license

- **NLTK**
  Website: http://www.nltk.org/

Code: https://github.com/nltk
Licence: Apache License, Version 2.0

References

[1] Horiguchi, Inoue, Hoshi, Okada, "GaChat:A chat system that displays online retrieval information in dialogue text"

[2] Abela, Cortis, SemChat: Extracting Personal Information from Chat Conversations

[3] Creswell, Schwartzmyer, Srihari, Information extraction for multi-participant, task-oriented, synchronous, computer mediated communication: a corpus study of chat data

[4] Tuulos, V.H., Combining Topic Models and Social Networks for Chat Data Mining

[5] Adams, P.H., Martell, C.H., Topic Detection and Extraction in Chat

[6] jQuery JavaScript Library, http://jquery.com/

[7] Bootstrap HTML5 & CSS3 framework v3.0.2 http://getbootstrap.com/

[8] scikit-learn, Machine Learning in Python, Python library

[9] Andrew, Nigam, A Comparison of Event Models for Naive Bayes Text Classification

[10] The XMPP Standards Foundation http://xmpp.org/

[11] Chat API, Facebook Developers https://developers.facebook.com/docs/chat/