Which Text-Mining Technique Would Detect Most Accurate User Frustration in Chats With Conversational Agents?

Hauke Hinrichs and Nguyen-Thinh Le
Humboldt-Universität zu Berlin
Germany
{hinricha, lenguyen}@informatik.hu-berlin.de

Conversational agents are getting ever more prevalent in online activities. There are many different approaches to measuring acceptance rate for such systems. In this paper, we explore the option of detecting user frustration in text-based user messages. Five text mining techniques (Decision Table Majority, Naive Bayes, Multilayer Perceptron, Sequential minimal optimisation, and K*) are compared in a supervised learning scenario using different quantifiable parameters. The comparison between these techniques shows that Sequential Minimal Optimisation is quickest and most accurate for detecting user frustration in text-based user messages.

Conversational agents, chat bots, text mining, supervised learning.

1. INTRODUCTION

Conversational agents (CA) are defined as systems that can keep a coherent conversation with human using different types of interfaces. The use cases of conversational agents are wide ranging from personal assistants on a cell phone over booking assistants on web sites for hotels and flights (Allen et al. 2001) to pure entertainment (Existor 2018).

Detecting user frustration in a conversation with a CA is useful for a number of reasons. CA developers can later review conversations where frustration is detected in order to improve the CA. In a dynamic scenario, the conversational agent could itself try to resolve the frustration upon detecting it, e.g., (Ramos 2017; van Eeuwen 2017) or pass the interaction to a human expert. Klein et al. (2002) found that users are more engaged with CA and the conversations last longer, if the user is not frustrated. Thus, detecting and reacting to frustration could vastly decrease the rate of aborted conversations. Additionally, Miner et al. (2016) found that users perceive negative statements as worse when coming from a CA in comparison to a human conversation partner.

Frustration is widely understood as not an emotion, but as a state that can result in an emotional reaction. Opinions differ regarding the resulting emotions. Dollard et al. (1939) claimed that frustration unavoidably results in either anger or aggression and the presence of these emotions always indicate underlying frustration. Bandura (1973) suggested that frustration is an individual reaction, varying from person to person depending on personal education and experience. Berkowitz (1989) later added depression and sadness as a possible result of frustration.

Text mining can be understood as a special type of data mining, through which non-trivial information is extracted from text data, whereas data mining occupies itself with extracting information from any type of data (Tan et al. 1999). Different attempts at detecting user emotion in various scenarios have been attempted, including deriving emotional state from peripheral information such as tone of voice and biological factors such as skin conductance (Greco et al. 2016). Kapoor et al. (2007) conducted experiments, in which participant's frustration was tracked using self-reports. This paper limits itself to textual information. The goal of this paper is to compare different text mining techniques regarding the accuracy in detecting user frustration. The resulting process and model after applying a text mining technique could be extracted and used in an existing conversational agent.

2. METHODOLOGY

In order to find the most fitting method for detecting user frustration in text-based user messages involving a CA, different text mining techniques are compared using a database of manually tagged chat

Published by BCS Learning and Development Ltd.
Proceedings of British HCI 2018. Belfast, UK.

lines. Additionally, a selection of pre-processing methods are compared in conjunction with each text mining technique. The overall process yielding a large number of models is shown in Figure 1.

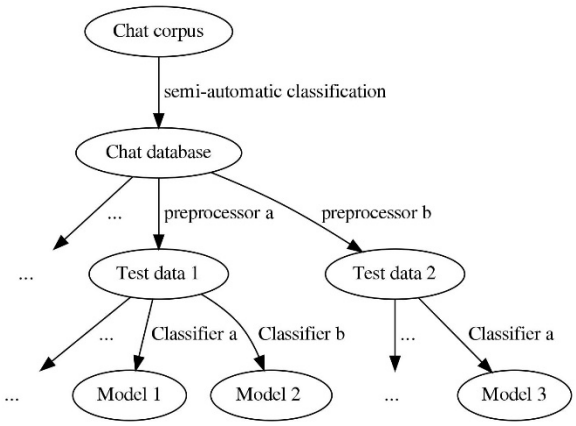

Figure 1. *Data flow and model creation process*

2.1 Data collection and coding

Based on the findings of Berkowitz (1989), we assume that a user is frustrated if either sadness or anger is visible in a text message. The presence of these emotions are deducted using the emotional lexicon EmoLex by Mohammad and Turney (2013). This lexicon maps more than 10,000 words to their respective associated emotions. The eight emotions used (anger, disgust, sadness, surprise, fear, trust, joy and anticipation) are taken from the basic levels of the wheel of emotions defined by Plutchik (2001). This mapping is used to assist the semi-automatic process of flagging the test data.

The chat excerpts are taken from publicly available corpora of conversations between humans and a conversational agent in English language (AI Research 2018). These chat excerpts are then flagged as ***frustrated*** or ***not frustrated*** based on the emotion determined by EmoLex using the words used in the message. This is carried out in a semi-automated manner to get a large body of test data in a reasonable time while also ensuring accuracy. All sentences are pre-flagged for confirmation or correction by a human using the mapping provided in EmoLex in a simple command line script. Flagging the data in this manner results in a data set with roughly 10% of the messages being annotated as ***frustrated***. This data (whose format consists of the message and a frustrated flag) is then pre-processed using different pre-processing methods.

2.2 Pre-processing

In addition to implementations of text mining techniques provided by the data mining software Weka University of Waikato (2018), this software also provides a wide variety of pre-processing methods. Tokenisers decompose an input test into a word vector that represents token appearances the text, either as a count or a boolean value. Stemmers break down words to some kind of root, so that different conjugations of the same root result in the same token. The word tokeniser, n-gram tokeniser and the Lovin's stemmer (Lovins 1968) are included in the pre-processing step in this study.

Additionally, adopting a different pre-processing technique, user messages are run through the NLTK developed by Loper and Bird (2002). In this process, negative sentence parts are marked as such in context so that the word vector contains negative words with a unique prefix, for example NEG_. Using this method, the sentence "*I don't find this funny*" would be turned into "*I don't NEG_find NEG_this NEG_funny*". With this method, it is possible for the text mining algorithm to differentiate between similar sounding statements more easily, because the sentence used in the example does not contain the token "funny" that could otherwise lead to the sentence being classified as containing positive emotions. After the pre-processing procedure, the test data consists of an input token vector determined by the tokeniser and the value for the ***frustrated*** flag.

2.3 Classifiers

As for the classifiers five different candidates are compared, all of which are included in Weka by default. These classifiers were chosen from the Weka library with the aim to be as different as possible to provide a broad overview over the possible choices.

The first and most simple one is the decision table. It composes a short list of rules during training, against which the test data is matched and classified. Rules are added until the improvement in accuracy falls below a pre-determined threshold. The Decision Table Majority (DTM) used in Weka falls back to the majority output in case no rule applies (Kohavi 1995).

Naive Bayes is a more complex and more widely used classifier, due to fast training and relatively high accuracy (McCallum et al. 1998). It calculates independent probabilities for input variables to cause changes to output variables during training and applies them on the test data.

A more complex (with respect to training procedure) classifier is the multilayer (MLP). It is an example of an artificial neural network with an input layer, any number of hidden layers and one output layer. Each layer can have any number of nodes, the values of which are propagated to any number of nodes in the next layer multiplied by the edge weight and added to a bias. During training the weights and biases are calculated using back propagation resulting in a quick classification (Pal & Mitra, 1992).

Sequential minimal optimisation (SMO) builds upon support vector machine (SVM), which represents

each test instance as an n-dimensional vector, n being the size of the input word vector. The resulting n-dimensional hyperplane is divided by an (n-1)-dimensional hyperplane while keeping the maximum distance to each test instance. Any new test vector is then classified based on which side of the dividing plane (Platt 1998).

K*, a special technique of Weka, is a lazy or instance based classifier, meaning that it takes zero time to train and compares each test instance against the training data during the classification. It uses a custom distance function to calculate the closest match (Aha et al. 1991). It is not a distance in the metric sense as it is not symmetrical and the distance of one instance to itself can be unequal 0. Cleary and Trigg (1995) claimed that it achieves better results over a wide variety of data sets than other instance based classifiers.

3. RESULTS

The accuracy of detecting user frustration is measured using F-measure and the Matthews correlation coefficient (*MCC*). *F* Measure is calculated using the Precision (*p*) and Recall (*r*) and ranges between 0 (worst) and 1 (best). However, F-Measure does not take into account the number of true negatives. Thus, MCC for binary (i.e., two-class) classifications can be used as a complementary quality measure. MCC takes true positive, true negative, false positive, false negative into account, resulting in a number between -1 and +1 with higher numbers denoting a better classification. MCC with value -1 indicates a total disagreement between expected results and observation. T_M represents the time it took to train the model, except for the classifier K*, where it denotes the time it takes to classify the test instances (since training is not required by instance based classifier K*). Thus, T_M cannot be directly compared between classifiers, but should rather convey a general sense of how quick each classifier performs given each data set.

Table 1. *Classification with word tokens and no stemming*

Class.	p	r	F	MCC	T_M
Bayes	0.538	0.304	0.389	**0.365**	0.26s
DTM	0.862	0.272	0.413	**0.462**	5.64s
MLP	0.133	0.337	0.191	**0.089**	4372s
SMO	0.769	0.435	0.556	**0.551**	0.36s
K*	0.882	0.326	0.476	**0.515**	19.24s

The results of classifying the data set with almost no pre-processing (Table 1) indicate that the complex classifier MLP is not necessarily the most accurate, with an MCC of 0.089. With the MCC being almost zero, it is not substantially better than a classifier that, for instance, just guesses based on the output distribution of the given training data. SMO and K* are the most accurate with MCCs of 0.551 and 0.515, respectively, while SMO is also among the quickest classifiers, taking only 0.26 seconds for training. Bayes is even quicker, taking 0.26 seconds, but not quite as accurate, being also beaten by DTM in terms of accuracy.

Table 2 shows that stemming improves classifier accuracy across the board. Compared to Table 1, the improvements regarding MCC range from 0.01 for K* to 0.357 for MLP. SMO is also the most accurate for this data set, with an MCC of 0.578, while sharing the first place for speed with 0.14 seconds, the same as Bayes. MLP profits most from, jumping from 0.089 (cf. Table 1) to an MCC of 0.446, beating Bayes, which scores an MCC of 0.427. All classifiers provide usable accuracy using these pre-processing methods. T_M stays roughly the same for each classifier as the input vector size is also about the same size, shrinking due to the stemming compared to the previous data set (cf. Table 1).

Table 2. *Classification using word tokens and stemming*

Class.	p	r	F	MCC	T_M
Bayes	0.633	0.337	0.440	**0.427**	0.14s
DTM	0.941	0.348	0.508	**0.553**	5.37s
MLP	0.737	0.304	0.431	**0.446**	3657s
SMO	0.782	0.467	0.585	**0.578**	0.14s
K*	0.886	0.337	0.488	**0.525**	18.1s

Table 3 shows that n-gram tokens provide a definitive benefit over word tokens but do not result in more accuracy than stemming word tokens. All classifiers yield a lower MCC compared to the data set with stemming and word tokens. The vastly larger input vector size (4734 values) due to n-gram generating more tokens leads to higher training times compared to the first data set (cf. Table 1). This is especially clear when looking at T_M for DTM, MLP and K* where it is an order of magnitude larger compared to both previously shown data sets. Bayes and SMO are not affected as much and are the fastest classifiers still for this data set, taking 0.76 and 0.29 seconds respectively (cf. Table 3).

Table 3. *Classification using n-gram tokens and no stemming*

Class.	p	r	F	MCC	T_M
Bayes	0.459	0.370	0.410	**0.363**	0.76s
DTM	0.926	0.272	0.420	**0.482**	39.06s
MLP	0.188	0.326	0.238	**0.152**	21263s
SMO	0.837	0.391	0.533	**0.548**	0.29s
K*	0.831	0.326	0.469	**0.498**	109.3s

Combining stemming and n-gram tokens does not provide a cumulative advantage over each pre-processing technique (i.e., either stemming or n-gram tokening) (cf. Table 4). The classifiers perform better than using the data set with n-gram tokens and no stemming, with the MCC improving between 0.008 and 0.101, but worse or no better than using the data with regular word tokens combined with stemming, compared to which the MCC drops up to 0.193. T_M stays roughly the same as for the not-stemmed n-gram tokens with the input vector token having about the same size with 4577 values.

Table 4. *Classification using n-gram tokens and stemming*

Class.	p	r	F	MCC	T_M
Bayes	0.493	0.402	0.443	**0.399**	0.69s
DTM	0.941	0.348	0.508	**0.553**	37.88s
MLP	0.319	0.315	0.317	**0.253**	20521s
SMO	0.848	0.424	0.565	**0.576**	0.29s
K*	0.857	0.326	0.472	**0.506**	102.7s

Results for using negation marking (cf. Section 2) are worse for every single classifier compared to each pre-processing method with respect to MCC. Based on Table 5, no classifier performs better than based on the data without negation marking. These numbers also show the flaw in using F-Measure as a sole performance indicator, as the F-Measure for MLP indicates positive performance, while the MCC shows that it provides almost no benefit over random guessing based on output distribution. T_M is roughly the same as in the first two data sets as the input vector is also about the same size.

Table 5. *Classification using negation marked input data*

Class.	p	r	F	MCC	Z_M
Bayes	0.464	0.283	0.351	**0.317**	0.17s
DTM	0.857	0.261	0.400	**0.451**	5.85s
MLP	0.100	0.239	0.141	**0.027**	5666s
SMO	0.704	0.413	0.521	**0.508**	0.16s
K*	0.900	0.293	0.443	**0.493**	20.1s

Table 6 confirms the MLP as the worst classifier of text mining techniques in this scenario with an MCC of only 0.194. SMO performs best with an MCC of 0.552 while also being the quickest, taking only 0.32 seconds followed by K* and DTM with an MCC of 0.507 and 0.5, respectively. Bayes stills performs better than MLP while also being a lot faster, scoring an MCC of 0.374 and taking 0.42 seconds.

Table 6. *Average results by classifier*

Class.	p	r	F	MCC	Z_M
Bayes	0.518	0.339	0.407	**0.374**	0.42s
DTM	0.905	0.300	0.450	**0.500**	19.07s
MLP	0.295	0.304	0.264	**0.194**	11096s
SMO	0.788	0.426	0.552	**0.552**	0.32s
K*	0.872	0.322	0.470	**0.507**	54.7s

Table 7: *Average results by input data set*

Data set	p	r	F	MCC
word tokens	0.763	0.334	0.458	**0.473**
w. t. & stemming	0.810	0.372	0.505	**0.521**
n-g. tokens	0.764	0.340	0.458	**0.473**
n-g. & stemming	0.785	0.375	0.497	**0.508**
neg. marking	0.731	0.313	0.429	**0.442**

Comparing the pre-processing techniques used on the different data sets, Table 7 shows the average results over data set with different pre-processing methods. The data set that produces the best single result, word tokens combined with stemming classified by SMO with an MCC of 0.578 (cf. Table 2), also performs best on average for all classifiers, reaching an MCC of 0.521 on average. Combining stemming and n-gram tokens results in the second best results on average with an MCC of 0.508, followed by not-stemmed data with n-gram tokens and word tokens, both with an MCC of 0.473. Negation marked input provides the worst accuracy of the pre-processing methods (MCC=0.442).

4. DISCUSSION

Surprisingly, the most complex and time intensive classifier does not provide the most accurate results for detecting frustration in text-based user messages. The bad performance of MLP is especially interesting as it is a widely used tool for complex tasks, for example image recognition and classification (Atkinson & Tatnall 1997). This might be a result of unfitting configuration for this task. Due to the time it takes to classify the multilayer perceptron and the large number of options Weka provides, it was not viable to find the optimal settings in a trial-and-error approach. The long training time might be improved by reducing the number of hidden nodes. SMO is by far the best classifier for the frustration detection as it requires no further configuration and is fast even when confronted with large data sets. K* did not perform much worse, but the time it takes to for classification due to its instance based architecture makes it not an ideal candidate. In terms of pre-processing, stemming proved to be the most useful tool with n-gram tokens only improving accuracy by a smaller amount while also causing higher complexity due to a larger input vector size.

5. CONCLUSIONS AND FUTURE WORK

The goal of this paper was to compare different text mining techniques regarding their usefulness for detecting user frustration in a conversation between human and a conversational agent. A few techniques proved to be of little use for this task. Marking negative words in the context of a sentence did not improve accuracy for any classifier. MLP showed to be too slow and inaccurate to be useful in a real time scenario. The Bayes approach delivers passable results but was surpassed by others in terms of speed. The SVM with the improvements provided by SMO and trained on stemmed input data showed to be highly accurate and quick in deciding whether the user is frustrated or not.

In future, the comparison could be repeated using a larger input data set or in a different language. Stop words were also not utilised because existing stop word lists are largely focused on classifying text by content and category, not by emotion, and would probably remove words from the messages that are useful for detecting emotion. Taking context into consideration could also improve accuracy as these input data sets were only being classified on their own with no regards to previous messages by either party of the conversation.

6. REFERENCES

AHA, D. W., D. KIBLER, AND M. K. ALBERT (1991): Instance-based learning algorithms. Machine learning, 6, 37–66.

AI RESEARCH (2018): Ai Research Homepage. http://a-i.com/, online, retrieved 13/06/2018.

ALLEN, J., G. FERGUSON, AND A. STENT (2001): An architecture for more realistic conversational systems. In Proceedings of the 6th international conference on Intelligent user interfaces, ACM, 1–8.

ATKINSON, P. M. AND A. TATNALL (1997): Introduction neural networks in remote sensing. International Journal of remote sensing, 18, 699–709.

BANDURA, A. (1973): Aggression: A social learning analysis. Prentice-Hall.

BERKOWITZ, L. (1989): Frustration-aggression hypothesis: Examination and reformulation. Psychological bulletin, 106, 59.

CLEARY, J. G. AND L. E. TRIGG (1995): K*: An instance-based learner using an entropic distance measure. In Machine Learning Proceedings 1995, Elsevier, 108–114.

DOLLARD, J., N. E. MILLER, L. W. DOOB, O. H. MOWRER, AND R. R. SEARS (1939): Frustration and aggression.

EXISTOR (2018): Cleverbot Homepage. http://www.cleverbot.com/, online, retrieved 13/06/2018.

GRECO, A., A. LANATA, L. CITI, N. VANELLO, G. VALENZA, AND E. P. SCILINGO (2016): Skin admittance measurement for emotion recognition: A study over frequency sweep. Electronics, 5, 46.

KAPOOR, A., W. BURLESON, AND R. W. PICARD (2007): Automatic prediction of frustration. International journal of human-computer studies, 65, 724–736.

KLEIN, J., Y. MOON, AND R. PICARD (2002): This computer responds to user frustration: Theory, design, and results. Interacting with Computers, 14, 119–140.

KOHAVI, R. (1995): The power of decision tables. In European conference on machine learning, Springer, 174–189.

LOPER, E. AND S. BIRD (2002): NLTK: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1, Association for Computational Linguistics, 63–70.

LOVINS, J. B. (1968): Development of a stemming algorithm. Mech. Translat. & Comp. Linguistics, 11, 22–31.

MCCALLUM, A., K. NIGAM, ET AL. (1998): A comparison of event models for naive Bayes text classification. In AAAI-98 workshop on learning for text categorization, 752, 41–48.

MINER, A., A. CHOW, S. ADLER, I. ZAITSEV, P. TERO, A. DARCY, AND A. PAEPCKE (2016): Conversational Agents and Mental Health: Theory-Informed Assessment of Language and Affect. In Proceedings of the 4th International Conference on Human Agent Interaction, ACM, 123–130.

MOHAMMAD, S. M. AND P. D. TURNEY (2013): Crowdsourcing a Word-Emotion Association Lexicon, 29, 436–465.

PAL, S. K. AND S. MITRA (1992): Multilayer perceptron, fuzzy sets, and classification. IEEE Transactions on neural networks, 3, 683–697.

PLATT, J. (1998): Sequential minimal optimization: A fast algorithm for training support vector machines.

PLUTCHIK, R. (2001): The Nature of Emotions. American scientist, 89, 344–350.

RAMOS, R. (2017): Screw the Turing Test - Chatbots don't need to act human. https://venturebeat.com/2017/02/03/screwthe-turing-test-chatbots-dont-need-to-act-humanonline, retrieved 30/05/2018.

TAN, A.-H. ET AL. (1999): Text mining: The state of the art and the challenges. In Proceedings of the PAKDD 1999 Workshop on Knowledge Discovery from Advanced Databases, 8, 65–70.

UNIVERSITY OF WAIKATO (2018): Weka project Page. http://www.cs.waikato.ac.nz/ml/weka/ online, retrieved 13/06/2018.

VAN EEUWEN, M. (2017): Mobile conversational commerce: messenger chatbots as the next interface between businesses and consumers.