Emoticon Prediction in Text

Iva Gornishka
iva.gornishka@student.uva.nl
10415548

Cassandra Loor
cassandra.loor@student.uva.nl
10624635

Finde Xumara
finde.findexumara@student.uva.nl
10690832

Arif Qodari
arif.qodari@gmail.com
10711996

Abstract

Emoticon prediction in text is a problem closely related to opinion mining, sentiment analysis and reputation extraction. This article discusses the technique of predicting emoticons in a sequence of messages which are related to one another. An emoticon is predicted for each message in the sequence by taking into account not only the message itself, but also the previous one. This lead to the use of a Hidden Markov Model and the Viterbi decoding algorithm.

1 Introduction

This report discusses emoticon prediction, based on a sequence of messages. In order to find messages that are related to one another and can transition in positivity to negativity, an entire conversation is observed. Each message is classified by not only considering the features of a single message, but also by considering the previous messages of the same author. This problem is a complex and interesting one, since it also takes into account the transitions from one state to another.

In order to find messages with such transitions, a corpora containing longer sequences of text messages are used - these might be conversations, such as an e-mail exchange or simply chat data, as long as the corpora contains a sufficient amount of emoticons. As emoticons appear in chat data more often, Ubuntu chat data [8] is used. Unfortunately, due to the lack of emoticons in this specific corpus, the emoticons of all sentences, which do not contain an emoticon, must also be determined. In order to relabel these sentences, another prediction model must be trained. In order to train this model, a corpus of short, unrelated Twitter [1] messages, containing emoticons, is used. This model is then used to relabel the Ubuntu chat data.

After the Ubuntu chat data is collected, it is first preprocessed. Then, appropriate features are extracted and a model is trained. The current report does not discuss the choice of features in detail. Basic features were chosen for this project without any intentional aim on improving this selection. Since this task on its own can take years, it is left as future work. However, suggestions on how to improve these features are mentioned in the following sections.

Three basic classes are used in order to classify the data. Table 1 presents these classes and the corresponding emoticons, contained in this type of data.

As table 1 shows, this selection of emoticons displays positive, negative and neutral emotions. This data can be split further into extremely positive (:D) and extremely negative (; ( ) emoticons for more specific classification of messages. However, this is beyond the scope of this paper.

This project is part of the Natural Language Processing 1 course taught at the University of Amsterdam in 2014-2015

Table 1: Data classes

DATA CLASS	**EXAMPLE EMOTICONS**
positive	:) :] =) :D
neutral	:| o_o
negative	:(:[=(;(

2 Problem

The intent of this research is to determine the transition of emoticons in a conversation. Such transitions are found in sequential messages. As stated before, chat data from Ubuntu is used to analyse transitions. This leads to the following problem:

> *How can emoticons be predicted in a sequence of messages, using the emoticon of the previous message?*

This leads to several obvious steps that must be taken. These are:

- Preprocess corpora
- Extract features
- Train a model
- Predict emoticons for unseen data

Relabelling the data

However, since the Ubuntu chat data contains mainly technical conversations, there appeared to be a significant lack of emoticons. This means that the majority of the messages were labelled as 'neutral', which in turn implies a terribly biased prediction performance. In order to use this data and test the prediction performance, the decision was made to relabel the data. For every 'neutral' message, an emoticon was predicted using a classifier trained on Twitter data and discussed later in section 3. This lead to the acquisition of data containing enough emoticons to properly train a model and predict new examples. In order to avoid such artificial relabelling of the data in other projects, corpora from a more appropriate source should be used - e.g. Skype conversation logs or any other group chat logs.

Preprocessing

As stated before, both corpora must be preprocessed. As the messages in both corpora are written by users in their spare time, there is a presence of slang, grammatical and spelling errors. There are also other sub-optimal aspects of the data that must be handled. After preprocessing, the corpora are ready for feature extraction. These subjects will be discussed later in this paper.

The models

In this research, Perceptron was used to predict emoticon for independent message. Two different kinds of perceptron were implemented: the average multiclass perceptron (AMP) and the multilayer perceptron (MLP). The MLP is more complex model but it has an advantage that it is possible to yield better results due to the presence of hidden layers. This research compares performance of both models to show which yields best results.

The model used to predict emoticons on sequences is the first-order Hidden Markov Model (HMM). The first-order HMM uses the emoticon of the previous message to predict the emoticon of the message that is being analysed.

It is expected that the two models will work separately. Even if the first model, used to relabel the sequence-based data, performs poorly, the performance of the HMM will not be affected. As the HMM is based on mislabelled data, it will learn these labels and continue building upon this. This will lead to sub-optimal results, overall, but the HMM will still perform well. Also, the sequence-based data in itself is slightly biased. As this data was collected from a support forum, it is expected that most data will be negative. This can lead to biased prediction. This must be prevented, by creating a balanced dataset of positive, negative and neutral data.

Balancing dataset for all classes is can be done easily for Twitter data but that would be difficult for Ubuntu chat dataset. As a solution, data selection technique was performed when building training set. It takes a sequence of messages based on the first class appears in the sequence. This technique makes the resulting training set has the same amount of data for all classes based on the first messages. However, this technique still does not fully guarantee the resulting training set is balance for all messages.

3 Emoticon Prediction in Single Message

This section discusses how Twitter data is used to train a classifier to predict emoticons basec on a single message.

3.1 Data

In order to collect data from Twitter, a crawler was used as discussed in [7]. The Twitter crawler searched for training and test data, based on emoticons. These messages ('tweets') contain much information - the name of the author, text, hyperlinks, hashtags (sequences of the form '#word_sequence'). As there are no official restrictions, messages often contain plenty of intentional or - more often - unintentional spelling and grammar errors. Messages also sometimes contain words from other languages, which makes preprocessing and analyzing the data even harder.

3.2 Preprocessing

Before any of the data can be used, the messages need to be preprocessed. This process includes three basic steps. Each one of them is discussed in more details in the following paragraphs.

Splitting the message

The message is first split into several components that will help in training. In this case, the text, hashtags and emoticons are extracted from the message. The existence of emoticon in a message can be used to label the message to a certain class (positive, negative, or neutral). While the text itself and the hashtags are used for feature extraction.

Spelling and grammar correction

Many messages contain spelling and grammar errors. In order to optimally use the Twitter data, all messages must be fixed spelling and grammar wise. There are several ways of performing these corrections, however these fall beyond the scope of this research. Therefore, names, links, emoticons and hashtags are disregarded when processing data, but the spelling and grammar errors remain. It is imperative to solve these errors in the future for optimal performance.

3.3 Model

As explained earlier, two models were implemented in order to classify single message into a certain class. A multiclass perceptron is an algorithm that allows supervised classification. As the name of the algorithm implies, the input does not have to be binary allowing input to be classified as one of many classes. As the Twitter data has to be classified as positive, neutral and negative before assigning an emoticon to the message, this algorithm does as desired.

Basically, a multiclass perceptron finds a border between two classes and draws a border between the two classes. By repeating this, several borders will appear between classes and are used to classify input. This leads to certain "gray"-areas, as different borders assign input data to different classes. In the case of the average multiclass perceptron, the average border of all is computed. This creates one border, without any "gray"-areas.

As for the multilayer perceptron, hidden layers are added to the perceptron. These hidden layers help classify the data more accurately. Unfortunately, this model is very sensitive to change. Therefore, tuning the model is very difficult. After many attempts and hours of tuning, the implementation of this model had to be aborted in order to conduct the research at hand. This has lead to the use of the AMP to label the sequential messages.

3.4 Features

Once the data is collected and preprocessed, its features are extracted. Selecting good features can significantly improve the performance in many cases, but this task is not a trivial one. Due to time constraints, certain basic features were selected to conduct this research. However, it is believed that future research can explore a bigger (or different) set of features. The selected features are based on previous work in this area [2]. Table 2 presents the features which were extracted and used for classification. Since most of the features' names are self-explanatory, only some of the features are discussed in details in the following paragraphs.

Table 2: Extracted features

FEATURE NAME	DESCRIPTION
words	total number of words in the text
positive words	total number of positive words in the text
negative words	total number of negative words in the text
positive words hashtags	total number of positive words in the hashtags
negative words hashtags	total number of negative words in the hashtags
uppercase words	number of words containing only uppercase letters
special punctuation	number of special punctuation marks such as '!' and '?'
adjectives	number of adjectives in the text

Extracting features such as **positive words** and **negative words** requires the usage of a predefined lists of 'positive' and 'negative' words, which were downloaded from [5]. Both these lists contain words that are positive or negative and are used to help predicting the emoticon of a short message. While implementing this aspect, the question of what effect extremely positive and extremely negative words have arose. In order to incorporate this idea, a different score might be used for the different subcategories of positive or negative words. Due to time constraints, this could not be researched. However, this is interesting to research in the future.

Features such as the number of **adjectives** were extracted by first performing part-of-speech (PoS) tagging on each sentence and then counting the adjectives. Due to spelling and grammar errors in the messages, PoS tagging does not always perform well. As a consequence, slight inaccuracies are possible for this feature, though not significant enough to negatively influence the performance.

3.5 Experiments and Results

As the AMP needs to be used to relabel the Ubuntu chat data, different experiments were run to test the performance. Sample results are shown in table 3 and are be discussed in detail in this section. Each test is run based on a standard set of 500 samples per class to ensure balance. 80% of the data is used for training and 20% is used as a test set to measure the performance of the model. In order to ensure a proper resemblance of test results, the results are averaged over 10 tests. Also, all features as declared earlier are used.

3.5.1 Prediction with varying number of classes

To test how well the perceptron predicts data on a varying number of classes, the algorithm was run while predicting multiple classes. Initially, it is interesting to see how well the data is classified using two classes; positive and negative. There is also a third class representing neutral data. There are also classes which are defined as extremely positive and extremely negative. In total, these are 15 classes. The performance of the model is also tested in this number of classes.

Classes	Data per class	Iterations	Train Accuracy	Test Accuracy
2	500	50	52.60	51.95
3	500	50	38.93	37.73
15	500	50	10.62	9.01

Table 3: AMP accuracy for different number of classes

Classes

Unsurprisingly, the results show that the more classes are used the worse the performance becomes.

In order to test the performance on two classes, the messages containing different'happy' and 'sad' emoticons were split into two general groups - 'negative' and 'positive' data. Since the data in these classes is significantly different, especially with respect to features such as count of positive or negative words, distinguishing differences seems to be much easier.

For testing the performance on three classes, data containing no emoticons or neutral emoticons were added to the training set as 'neutral' data. In this case the performance dropped dramatically. The lack of an emoticon in a message does not necessarily mean that this message is not a positive or a negative one, which is reflected in the results. For example, some people do not use emoticons to express their emotions, but hashtags instead. Thus, neutral data is often classified mistakenly as positive or negative or vice versa, which significantly decreases the overall performance of the perceptron.

For testing purposes, the data was eventually split into 15 different classes, each of them containing messages with a different emoticon - i.e. messages containing emoticons such as ':)' and ':D' were not both considered 'positive' anymore, but treated as different classes instead. This has lead to very small datasets for certain classes. This leads to bias of the larger classes, mislabelling most data.

Features

The effect of different features was also explored. Each one of the features was being excluded and prediction performance was tested using the rest of the features. In order to see clear results, two classes were used to classify the data. These results are displayed in table 4.

Features Excluded	Data per class	Iterations	Train Accuracy	Test Accuracy
None	500	50	52.60	51.95
Positive/negative words	500	50	53.88	52.90
Positive/negative hashtags	500	50	54.40	53.70
Amount of words	500	50	51.78	50.90
Uppercase words	500	50	51.51	49.65
Special punctuation	500	50	51.91	51.25
Adjectives	500	50	52.39	50.25

Table 4: AMP accuracy

Tests show that some of the features are not meaningful enough and including them only decreases the performance. For example, the use of hashtags should improve the performance, but it appears that when emoticons are used, hashtags aren't. Therefore, the features that represent the hashtags are mainly empty. This does not help distinguish positive data from negative data and should be excluded from the feature set. As this model is used to label data which does not contain hashtags, this should not make a difference. Also, the sequence-based data hardly contains words represented solely in uppercase. Therefore, this feature can also be excluded from the feature set. This leads to the following results

Features Excluded	Data per class	Iterations	Train Accuracy	Test Accuracy
Positive/negative hashtags and Uppercase words	500	50	56.96	56.25

Table 5: AMP optimal accuracy

Excluding the positive and negative hashtags, as well as the uppercase words, yield the best results and will be used for relabelling the sequence-based data.

3.6 Dataset size

This section discusses the effect of different dataset sizes on the AMP implementation. The results are presented in table 6. As hashtags are hardly used, this feature was turned off to research the sizes of the dataset. This will create less similarity between the classes.

Table 6 indicates that the more data is used, the more the model generalizes. However, the performance of the model remains poor. This happens due to poorly chosen features. Also, the model seems unstable as there is no clear increase

Classes	Samples per class	Clusters	Train Accuracy (%)	Test Accuracy (%)
2	50	50	59.75	48.00
2	100	50	55.75	52.25
2	200	50	55.19	53.63
2	300	50	54.65	53.75
2	400	50	54.55	52.56
2	500	50	55.78	54.80
2	1000	50	53.47	53.80
2	2500	50	54.68	54.82

Table 6: AMP size of dataset

and decrease in performance. However, as the difference in performance is smallest when using a dataset of 2500 samples per class, this indicates that generalization is being achieved.

4 Emoticon Prediction in Sequential Message

4.1 Data

In this part of the project the Ubuntu Chat Corpus was used as training data [8]. This corpus consists of archived chat logs from Ubuntu's Internet Relay Chat technical support channels. The logs contain continuous conversations, where numerous authors take part in the conversation. Some authors only post few messages, whereas others participate more actively in the conversations. As discussed earlier, this data does not contain enough emoticons, hence, relabeling the whole corpus was necessary. After an emoticon was predicted for each sentence, the data was ready to be used to feature extraction, training and testing.

4.2 Preprocessing

Since the Ubuntu Chat corpus is well structured and hardly contains any hashtags, hyperlinks, capitalized sentences or other special message parts appear, preprocessing the data would not significantly increase the performance. This lead to the decision that the preprocessing step for this data set would be omitted for the purpose of this project.

4.3 Model

A first-order Hidden Markov Model was chosen for the purpose of this project. This section summarizes the steps which were taken in order to train the model and predict classes for unseen examples afterwards.

As explained earlier, after the data has been pre-processed, a feature vector is extracted for each example message. This feature vector is similar to the feature vector extracted from the Twitter data, excluding hashtags and uppercase words. Thus, the training data for the model consists of pair (X, y), where X is a feature vector and y is the corresponding class label. The next step is to compute the transition and emission probabilities, i.e. the probability of transitioning from one class state to another and the probabilities of the different feature vectors given a class state.

Consider the following: the probability of observing a ”negative” message immediately after a ”neutral” message has been observed, is a transition probability. Furthermore, the probability of the feature vector of the message 'I need to get some sleep now', given the neutral class, is an emission probability. This means that the following general formulas can be used in order to calculate these probabilities:

$$\mathcal{P}(v|u) = \frac{c(u,v)}{c(u)}$$

and

$$\mathcal{P}(X|y) = \frac{c(y \rightarrow X)}{c(y)}$$

Where $\mathcal{P}(v|u)$ is the transition probability from class u to v. The amount of times a class is followed by another class is observed in the training data and can be determined by $c(u, v)$. For example, $c(positive, negative)$ is the amount of times class $positive$ followed by class $negative$ is observed in a sequence in the training data.

Similarly, $c(u)$ defines the amount of times the class u is seen in the training data. For example, $c(positive)$ is the number of times class $positive$ is seen in the training data.

Moreover, $\mathcal{P}(X|y)$ is the emission probability of feature vector X given class y. The amount of times class y is seen, paired with feature vector X, can be defined as $c(y \rightarrow X)$.

Since the total amount of feature vectors X, which can be observed, is too large, not all possible vectors would be observed in the training data. This would make predicting unseen messages impossible, since no emission probabilities would be present for these vectors. In order to avoid this, the training data is first clustered. Emission probabilities are computed for each cluster. Then, given an unseen message, its feature vector is assigned to a cluster. Finally, a label is predicted for this message.

4.4 Features

The corpus of this section does not contain hashtags and hardly contains uppercase words. Therefore, these features are excluded from the feature set. However, all other features described in the previous section are considered.

4.5 Experiments and Results

Different experiments were run to test the performance of the Hidden Markov Model. Sample results are shown in tables and are discussed in more detail in this section. Table 7 shows the accuracy for two classes (positive and negative) and all three classes (including neutral). The model was trained on 90% of the selected data and tested on the 10% that is left. Each test is run on all data available, unless specified otherwise. The amount of clusters is set to 50.

Classes	Features	Clusters	Train Accuracy (%)	Test Accuracy (%)
2	all	50	73.65	72.66
3	all	50	39.23	38.37

Table 7: HMM accuracy

Classes

Table 7 shows that the use of two classes yields best results. However, this happens due to the fact that the results are biased. As the chat data comes from a support forum, most messages are indeed negative or neutral. As the emission and transition values are calculated by counting labels, the HMM itself will most likely transition to negative or neutral. This also leads to few data labelled as positive always being correct. However, this does not happen often, due to the aforementioned calculations. Finding a better way to select data to train the model on will be imperative in the future in order to produce better results. As the data of three classes is least biased, this will be used in further testing.

Features

As with the AMP, the effect of the features on the HMM were tested. As this model trains on a sequence of messages, it could be that the features have a different effect.

Features Excluded	Classes	Iterations	Train Accuracy (%)	Test Accuracy (%)
None	3	50	39.23	38.37
Positive/negative words	3	50	38.69	38.62
Amount of words	3	50	38.61	38.02
Special punctuation	3	50	38.81	38.25
Adjectives	3	50	38.66	37.73

Table 8: AMP accuracy

Table 8 shows that the exclusion of more features shows no improvement. This confirms the suspicion of poorly selected features. However, this also does confirm that the current selection of features works optimal and should be used for further testing.

Amount of clusters

Different amounts of clusters were used to test their effect on the model. The results are displayed in table 9.

Classes	Clusters	Train Accuracy (%)	Test Accuracy (%)
3	10	38.27	37.94
3	20	38.53	38.36
3	30	39.00	37.45
3	40	38.99	38.22
3	50	39.23	38.37

Table 9: HMM amount of clusters

The results in table 9 shows that an increasing amount of clusters yields varying results. This could mean that the model overfits. However, there is not much difference between implementing few clusters and many clusters. The fact that the model does not yield good results can be attributed to the fact that the selected features are not sufficient. The minimal differences between the results leads to the suspicion that there is a bug in the implementation. However, due to time constraints, this bug was not identified and fixed.

4.6 Dataset size

This section discusses the effect of different dataset sizes on the HMM implementation. The results are displayed table 10.

Classes	Data per class	Clusters	Train Accuracy (%)	Test Accuracy (%)
3	50	50	45.17	39.65
3	100	50	41.90	38.76
3	200	50	40.91	38.48
3	300	50	40.67	37.53
3	400	50	40.41	37.34
3	500	50	40.32	36.67
3	1000	50	39.29	36.63
3	2500	50	39.00	36.14

Table 10: HMM dataset size

Table 10 indicates that when least data is used, the model performs best. There is a significant change in the performance when testing on the training set. However, there is a smaller change in performance when testing on the test set. This leads to the conclusion that the model overfits quickly. As the data is balanced with few data, the performance is expected to perform poorly on all levels. Given that the model actually yields best results with little training, this confirms that there is a bug in the implementation. However, due to time constraints, this cannot be found and fixed.

5 Conclusion

The selected corpora for this research was not sufficient and needed labelling in order to find the positivity, negativity or neutrality of a message. This means that the selected corpora was not sufficient for the research at hand and must be selected better in the future.

The preprocessing of the corpora was effective, but can be improved in the future. Minimal correction of grammar- and spelling errors was sufficient enough to find positivity, negativity or neutrality in a message. Perhaps when extreme positivity and negativity are also taken into account, these corrections need to be more specific. For the this research, this was sufficient.

All tests have shown that the selected features help predict emoticons. However, when excluding features, no exclusion lead to a dramatic rise or drop in accuracy. This means that the selected features are not sufficient and other and additional features must be determined for better performance.

When selecting the amount of clusters to determine the transition and emission probabilities, an increasing amount of clusters leads to worse results. It appears as though the model overfits. However, as the best performance and the worst performance with a certain amount of clusters hardly seems to make a difference, it is believed that there is a bug in the the implementation. Due to time constrains, it isn't possible to find this bug and to solve this problem.

The size of the dataset shows to have an impact on the AMP, as no conclusions can be drawn from the HMM tests. As long as the dataset is large, but balanced, generalization should take place. This will lead to a much better model.

6 Future work

6.1 Corpus selection

As discussed in section 2, the Ubuntu Chat Corpora does not contain a sufficient amount of emoticons for training. It has not been proven whether the artificial relabelling of the data has had an impact on the performance of the Hidden Markov Model, but it is believed that tests should be also performed using another corpora in order to compare the performance. Also, this alternate dataset should be balanced in such a way that there are equal transitions between positive, negative and neutral. This will lead to more general transition and emission probabilities. There sequence-based corpora that can be used. Facebook and Skype often contains the information that is necessary as stated here. However, this is not open and free to use, which makes finding a sufficient corpora difficult.

6.2 Feature selection

As mentioned earlier in this paper, feature selection is crucial for the emoticon prediction in text, just like it is in every prediction problem. It is believed that further exploration of the feature space could lead to much better results. A bigger set of features is discussed in [2]. Although a different type of data is used for this research, some of the discussed features might be applicable in other settings too. Last but not least, a set of features for sentiment analysis in Twitter data is discussed in [1].

6.3 Model selection

For the purpose of this project an Average Multiclass Perceptron (AMP) and a Hidden Markov Model (HMM) were selected as most appropriate. It has not been tested whether different models yield better performance, hence a future work in the area might include exploring different models. Different sources, such as [4], suggest using an SVM or Naive Bayes instead of the AMP for emotion prediction, and could be also appropriate for emoticon prediction. Furthermore, using a second order HMM could improve the performance for predicting sequences.

Team responsibilities

The four authors contributed equally to this project.

7 Software packages

Twitter data crawler

Tweepy was used to extract data from Twitter. More information about Tweepy can be found at `http://www.tweepy.org`.

Ubuntu Chat Corpus

The Ubuntu Char Corpus was used in this project as train and test data. More information about the Ubuntu Chat Corpus might be found at `http://daviduthus.org/UCC/`. Furthermore, details about the corpus might be also found in [8].

Natural Language Toolkit (NLTK)

The Natural Language Toolkit for Python was used in this project during the feature extraction stage. More information about it might be found on `http://www.nltk.org/`. Furthermore, [3] provides a practical introduction to programming for language processing.

References

[1] Apoorv Agarwal et al. “Sentiment analysis of twitter data”. In: *Proceedings of the Workshop on Languages in Social Media*. Association for Computational Linguistics. 2011, pp. 30–38.

[2] Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. “Emotions from text: machine learning for text-based emotion prediction”. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2005, pp. 579–586.

[3] Edward Loper Bird Steven and Ewan Klein. *Natural Language Processing with Python*. 2009, publisher =.

[4] Taner Danisman and Adil Alpkocak. “Feeler: Emotion classification of text using vector space model”. In: *AISB 2008 Convention Communication, Interaction and Social Intelligence*. Vol. 1. 2008, p. 53.

[5] Minqing Hu and Bing Liu. “Mining and summarizing customer reviews”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 168–177.

[6] Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. “NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets”. In: *arXiv preprint arXiv:1308.6242* (2013).

[7] Eyal Sagi and Morteza Dehghani. “Moral Rhetoric in Twitter: A Case Study of the US Federal Shutdown of 2013”. In: ().

[8] David Uthus and David Aha. *The Ubuntu Chat Corpus for Multiparticipant Chat Analysis*. 2013.