Natural Language Processing
Emoticon Prediction

By: Gornishka, Loor, Qodari, Xumara

December 3, 2014

Contents

1 Abstract

This report contains the implementation of an emoticon predictor, based on text. This is a project within Natural Language Processing (NLP), meaning many resources within the NLP community were consulted. This report discusses two different ways of prediction emotions. In the first case, emoticons are predicted based on a single short message. This lead to the use of an average multiclass perceptron. In the second case, chat data is used to predict emoticons, based not only on the current message, but also on the previous ones as well. This lead to the use of a Hidden Markov Model and the Viterbi decoding algorithm.

2 Introduction

3 Problem

As stated in the introduction, this report discusses two related, but different problems - emoticon prediction based on a single short message and based on a sequence of messages. In the first problem, some amount of unrelated short messages should be observed, together with the contained emoticons, a classifier should be trained and at the end, given a new message the corresponding emotion should be predicted. This means that the messages might (and should) be written by different authors in order to achieve good generalization. In the second problem, a whole conversation should be observed and the classification should be done not only by considering the features of a single message, but instead, by also considering the previous messages of the same user. This problem is a more complex and interesting one, since it also takes into account the transitions from one sentiment state to another.

As a natural consequence of the above statements of the two problems, two different data sets should be used for them. For the first one it might be any corpora of text messages, containing a decent amount of emoticons. Such corpora might contain for example any unrelated text messages from social media. For the current research, a Twitter data was chosen. For the second problem, a corpora containing longer sequences of text messages should be used - these might be different conversations, such as an e-mail exchange or just chat data, as long as they contain sufficient amount of emoticons. For the current research an Ubuntu chat data was chosen.

After the data is collected, it is first preprocessed. Afterwards appropriate features are extracted and a model is trained. The current report does not discuss the choice of features in detail. Some basic feature were chosen for this project without intentional aim on improving them. Since this task on its own deserves a lot of time and effort, it is left as future work and suggestions about it are mentioned in the following sections.

. . . .

4 Twitter emoticon prediction

In this part of the project Twitter data was used in order to train a classifier for predicting emoticons. Three basic classes were used in order to classify the data:

- **positive** - data containing 'positive' emoticons (expressing positive emotions) such as ':)', ':]', '=)', or ':D'
- **neutral** - data containing no emoticons or neutral emoticons like ':|' or 'o_o'
- **negative** - data containing 'negative' emoticons (expressing negative emotions) such as ':(', ':[', '=(' or ';('

wijzers

4.1 Data

In order to collect data from Twitter, a crawler was used. 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'). Since 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.

4.2 Preprocessing

Before any of the data can be used, the messages need to be preprocessed. This process includes the following steps:

- **Splitting the message** into different components - text, hashtags and emoticons.
- **Spelling check and correction**
- **Grammar check and correction**

4.3 Model

A multi-class perceptron was chosen as most appropriate for the purpose of this project.

4.4 Features

The following features were extracted from the data and used for classification:

- **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. Both these lists contain ......

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

4.5 Experiments and Results

5 Ubuntu emoticon prediction

5.1 Data

5.2 Preprocessing

Before any of the data can be used, the messages need to be preprocessed.

5.3 Model

A Hidden Markov Model was chosen as post appropriate 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. 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 a simple example: the probability of observing a 'sad' massage right 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 might be used in order to calculate these probabilities:

$$\mathcal{P}(\textit{transitioning from class y' to class y"}) = \frac{\textit{\# of times class y" is observed after class y'}}{\textit{total \# of training examples}}$$

$$\mathcal{P}(\textit{feature vector } X \textit{ given class } y) = \frac{\textit{\# of times the pair } (X, y) \textit{ occurs in the training data}}{\textit{total \# of training examples from class } y}$$

Since the total number of feature vectors X which can be observed is too big, 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 issue, the following solution is implemented: the training data is first clustered, emission probabilities are computed for each cluster, then given an unseen message, its feature vector is first assigned to a cluster and finally, a label is predicted for this message.

5.4 Features

The features used in this part of the project correspond to the ones already discussed in section 4.4. Note that some features, such as numbers of positive or negative words in hashtags are not applicable in the current setting, since the data does not contain hashtags. Thus, such features have not been considered.

5.5 Experiments and Results

6 Conclusion

7 Team responsibilities

Files attached

- Code

Sources

- Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. OReilly Media Inc.
- Author: Ryan Kelly, Home Page: https://pythonhosted.org/pyenchant/

I'm gonna cite something here [1] to see how it works ;)

References

[1] George D. Greenwade. “The Comprehensive Tex Archive Network (CTAN)”. In: *TUGBoat* 14.3 (1993), pp. 342–351.

- Some paper