Emoticon Prediction in Text

[1]

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

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

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

Arif Qodari
arif.qodari@gmail.com
ID

Abstract

Emoticon prediction in text is a problem closely related to opinion mining, sentiment analysis and reputation extraction as it heavily relates to emotion prediction. This article predicts emoticons, using two different techniques. 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, emoticons are predicted, using messages posted in the past. This leads to the previous message also determining the emotion of the message analyzed. This lead to the use of a Hidden Markov Model and the Viterbi decoding algorithm.

1 Introduction

This report discusses two related, but essentially different problems - emoticon prediction based on a single short message and based on a sequence of messages. In the first problem, short, unrelated messages are observed. Their emoticons are used to determine whether a message is positive or negative. Using this information, a is trained. Eventually, a new message can be passed and the corresponding emotion is predicted. In order to determine the sentiment of a message optimally and achieve generalization, messages of different authors are collected and used to train the classifier. In the second problem, 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 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, different data sets are used to approach each problem. For the first problem, any corpora of text messages containing emoticons can be used. Such corpora can contain unrelated messages. In order to find short unrelated messages, Twitter data is used. For the second problem, 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 is used.

After the 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 feature were chosen for this project without intentional aim on improving them. Since this task on its own can take up years, it is left as future work. However, suggestions about improving features is mentioned in the following sections.

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

In order to classify the data, classes must be determined. Three classes are used to classify the data. Table 1 displays these classes and the corresponding emoticons, contained in this type of data.

Table 1: Data classes

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

As table 1 shows, known emoticons display positive, negative and neutral emotions. However, 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 and is further mentioned in future work. . . . . .

2 Problem

3 Emoticon prediction based on short messages

This section discusses how Twitter data is used to train a classifier to predict emoticons.

3.1 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'). 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, itself, is split into different components that will help in training. In this case, the text, hashtags and emoticons are extracted. Emoticons in messages are used to add emotion, but not text. The text itself can be used to determine whether a message is positive, negative or neutral. Therefore, the text without the emoticons can be used to determine which emoticon can be assigned to a certain message. Also, the hashtags can be used to further determine the emotion of the message. Hashtags often contain extra information, such as 'sad' or 'ecstatic'. This can also help determine the emotion of a message. In this case, however, the hashtags were not used. Therefore, it is important to research the effect of this feature in the future.

Spelling and grammar correction

Short messages are often posted as a reaction or something to share with friends quickly. This means that 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

A message can be classified as a positive, negative or neutral emotion. In order to classify these messages, an averaged multi-class perceptron was implemented. 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 multiclass perceptron, the average border of all is computed. This creates one border, without any ”gray”-areas.

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 research conducted in the future can explore a bigger (or different) set of features. Table 2 presents the features which were extracted and used for classification. Since most of the features' names are self-explanatory, only the 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 SOME REFERENCE. Both these lists contain words that are positive or negative and are used to classify the emotion of a short message. While implementing this aspect, the question of what effect extremely positive and extremely negative words have arose. 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 in the tagged sentence. 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

4 Ubuntu emoticon prediction

4.1 Data

4.2 Preprocessing

Before any of the data can be used, the messages need to be preprocessed. This is done again in the way explained in section 3.2.

4.3 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.

4.4 Features

The features used in this part of the project correspond to the ones already discussed in section 3.4. 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.

4.5 Experiments and Results

5 Conclusion

6 Future work

Team responsibilities

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.

GENERAL THINGS AND INFO FROM THE TEMPLATE AFTER HERE!!!!

7 Submission of papers to NIPS 2014

NIPS requires electronic submissions. The electronic submission site is

http://papers.nips.cc

Please read carefully the instructions below, and follow them faithfully.

7.1 Style

Papers to be submitted to NIPS 2014 must be prepared according to the instructions presented here. Papers may be only up to eight pages long, including figures. Since 2009 an additional ninth page *containing only cited references* is allowed. Papers that exceed nine pages will not be reviewed, or in any other way considered for presentation at the conference.

Please note that this year we have introduced automatic line number generation into the style file (for LaTeX 2_ε and Word versions). This is to help reviewers refer to specific lines of the paper when they make their comments. Please do NOT refer to these line numbers in your paper as they will be removed from the style file for the final version of accepted papers.

The margins in 2014 are the same as since 2007, which allow for $\approx 15\%$ more words in the paper compared to earlier years. We are also again using double-blind reviewing. Both of these require the use of new style files.

Authors are required to use the NIPS LaTeX style files obtainable at the NIPS website as indicated below. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

7.2 Retrieval of style files

The style files for NIPS and other conference information are available on the World Wide Web at

http://www.nips.cc/

The file `nips2014.pdf` contains these instructions and illustrates the various formatting requirements your NIPS paper must satisfy. LaTeX users can choose between two style files: `nips11submit_09.sty` (to be used with LaTeX version 2.09) and `nips11submit_e.sty` (to be used with LaTeX2e). The file `nips2014.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own. The file `nips2014.rtf` is provided as a shell for MS Word users.

The formatting instructions contained in these style files are summarized in sections 8, 9, and 10 below.

8 General formatting instructions

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing of 11 points. Times New Roman is the preferred typeface throughout. Paragraphs are separated by 1/2 line space, with no indentation.

Paper title is 17 point, initial caps/lower case, bold, centered between 2 horizontal rules. Top rule is 4 points thick and bottom rule is 1 point thick. Allow 1/4 inch space above and below title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in section 10 regarding figures, tables, acknowledgments, and references.

9 Headings: first level

First level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 12. One line space before the first level heading and 1/2 line space after the first level heading.

9.1 Headings: second level

Second level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the second level heading and 1/2 line space after the second level heading.

9.1.1 Headings: third level

Third level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the third level heading and 1/2 line space after the third level heading.

10 Citations, figures, tables, references

These instructions apply to everyone, regardless of the formatter being used.

10.1 Citations within the text

Citations within the text should be numbered consecutively. The corresponding number is to appear enclosed in square brackets, such as [1] or [2]-[5]. The corresponding references are to be listed in the same order at the end of the paper, in the **References** section. (Note: the standard BIBTEX style `unsrt` produces this.) As to the format of the references themselves, any style is acceptable as long as it is used consistently.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]", not "In our previous work [4]". If you cite your other papers that are not widely available (e.g. a journal paper under review), use anonymous author names in the citation, e.g. an author of the form "A. Anonymous".

10.2 Footnotes

Indicate footnotes with a number[2] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).[3]

10.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction; art work should not be hand-drawn. The figure number and caption always appear after the figure. Place one line space before the figure caption, and one line space after the figure. The figure caption is lower case (except for first word and proper nouns); figures are numbered consecutively.

Make sure the figure caption does not get separated from the figure. Leave sufficient space to avoid splitting the figure and figure caption.

You may use color figures. However, it is best for the figure captions and the paper body to make sense if the paper is printed either in black/white or in color.

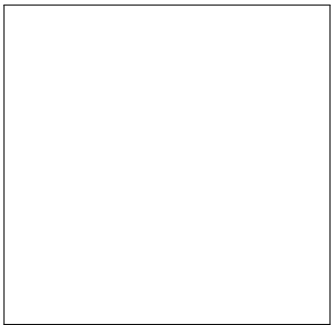

Figure 1: Sample figure caption.

[2] Sample of the first footnote

[3] Sample of the second footnote

Table 3: Sample table title

PART	DESCRIPTION
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

10.4 Tables

All tables must be centered, neat, clean and legible. Do not use hand-drawn tables. The table number and title always appear before the table. See Table 3.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

11 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

12 Preparing PostScript or PDF files

Please prepare PostScript or PDF files with paper size "US Letter", and not, for example, "A4". The -t letter option on dvips will produce US Letter files.

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`
- LaTeX users:
  - Consider directly generating PDF files using `pdflatex` (especially if you are a MiKTeX user). PDF figures must be substituted for EPS figures, however.
  - Otherwise, please generate your PostScript and PDF files with the following commands:

    ```
    dvips mypaper.dvi -t letter -Ppdf -G0 -o mypaper.ps
    ps2pdf mypaper.ps mypaper.pdf
    ```

    Check that the PDF files only contains Type 1 fonts.
  - xfig "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
  - The `\bbold` package almost always uses bitmap fonts. You can try the equivalent AMS Fonts with command

    ```
    \usepackage[psamsfonts]{amssymb}
    ```

    or use the following workaround for reals, natural and complex:

    ```
    \newcommand{\RR}{I\!\!R} %real numbers
    \newcommand{\Nat}{I\!\!N} %natural numbers
    \newcommand{\CC}{I\!\!\!\!C} %complex numbers
    ```
  - Sometimes the problematic fonts are used in figures included in LaTeX files. The ghostscript program `eps2eps` is the simplest way to clean such figures. For black and white figures, slightly better results can be achieved with program `potrace`.

- MSWord and Windows users (via PDF file):
  - Install the Microsoft Save as PDF Office 2007 Add-in from `http://www.microsoft.com/downloads/details.aspx?displaylang=en\&familyid=4d951911-3e7e-4ae6-b059-a2e79ed87041`
  - Select "Save or Publish to PDF" from the Office or File menu
- MSWord and Mac OS X users (via PDF file):
  - From the print menu, click the PDF drop-down box, and select "Save as PDF..."
- MSWord and Windows users (via PS file):
  - To create a new printer on your computer, install the AdobePS printer driver and the Adobe Distiller PPD file from `http://www.adobe.com/support/downloads/detail.jsp?ftpID=204` *Note:* You must reboot your PC after installing the AdobePS driver for it to take effect.
  - To produce the ps file, select "Print" from the MS app, choose the installed AdobePS printer, click on "Properties", click on "Advanced."
  - Set "TrueType Font" to be "Download as Softfont"
  - Open the "PostScript Options" folder
  - Select "PostScript Output Option" to be "Optimize for Portability"
  - Select "TrueType Font Download Option" to be "Outline"
  - Select "Send PostScript Error Handler" to be "No"
  - Click "OK" three times, print your file.
  - Now, use Adobe Acrobat Distiller or ps2pdf to create a PDF file from the PS file. In Acrobat, check the option "Embed all fonts" if applicable.

If your file contains Type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

12.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the graphicx package. Always specify the figure width as a multiple of the line width as in the example below using .eps graphics

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.eps}
```

or

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

for .pdf graphics. See section 4.4 in the graphics bundle documentation (`http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

References follow the acknowledgments. Use unnumbered third level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to 'small' (9-point) when listing the references. **Remember that this year you can use a ninth page as long as it contains *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609-616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer-Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.