Automated Event Suggestion using Natural Language Processing Model

Rylan Marianchuk, Lisandro Mayancela, Murori Mutuma
Summer 2020

Google SPS: "Software Product Sprint"

ABSTRACT: We develop an elementary classification algorithm that considers as input messages within a group chat and determines, as output, if users are attempting to plan an event. Two methods are used to implement the classifier: (1) language feature extraction and (2) online learning with feedback. In the case our classifier returns positive for an event, a google calendar link is given to the group chat as output − this consists of relevant data extracted from previous messages and group chat members which includes time, date, title, and participant email addresses of the event. Given the classifier's speed and our attempts to reduce its false positives and false negatives, our solution provided through google calendar integration is practical in comparison to the tedious task of scheduling a (remote or not) event for a large quantity of participants.

AIM

Given the short two week span for our software product, we look to implement an elementary classifier of natural language messages with brevity. Our classifier will read messages within a group chat and output a google calendar link when it detects users, through their natural language, are speculating about meeting for an event. **The end goal** is to build a robust classifier, that is, an algorithm with low numbers of false negative and false positive classification errors.

METHODS

It would seem salient that a supervised learning algorithm could perform well for the binary classification task of natural language sentences. Predetermined labelled instances of messages

could be fed to a classifier to learn the classification boundary in its latent space. Unfortunately this cannot be accomplished, since from quick searches for data, there does not exist natural language messages labelled in the manner of our interest.

Our alternative naive unsupervised solution is to represent a message as a feature vector with predetermined components of language relevant to event suggestion. Additionally we add weights to these dimensions to scale their importance in classification.

Relevant Features

Each one of a, b, c, d, e are independent features of a natural language message that we have chosen to be of interest in event detection:

a. Time
   Does there exist a numerical quantity denoting a time in the message?
   Does 'am' or 'pm' occur in the message?
   Does a weekday occur in the message?

b. Verbs
   Using predetermined lexicon: "discuss", "meet", "talk", "plan", "hash", "review", in addition to their present tense: "discussing", "meeting", etc.

c. Question
   Is the message posed as a question?
   The existence of a question mark may imply the user is asking for an event to be scheduled.

d. True Negative Similarity
   When feedback (labelled messages of both negative and positive instances) is given to the model through its usage over time, this dimension will measure the degree to which a new instance is similar in semantic meaning to previously kept negative examples. The degree of similarity is measured using a Doc2Vec model [3]. (See section on "False Positive and Negative Reincorporation Learning")

e. True Positive Similarity
   Similar to above, this dimension uses past learned examples to quantify a new example's semantic similarity using a Doc2Vec model [3].

Message Representation

We consider a single natural language message as a vector, with each dimension as a real number representing one of the aforementioned chosen fundamental features. The alpha vector is the message scaling parameters to be optimized by the model

$$\mathcal{M} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} \qquad \mathcal{M}_c = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix}$$

The magnitude of the message feature vector is used for the final step in classification, represented by the formula below:

$$||\mathcal{M}|| = \sqrt{\sum_{i=1}^{5} \alpha_i \left(f_i\right)^2}$$

Classification Boundary

Although a hyperplane may be used (see section 4.5 at [2]) for a binary classification boundary, I chose to simply use a hard-coded threshold value which is to be compared to a message vector's magnitude for classification. This was because (a) we do not have labelled training examples to converge on the threshold parameter through learning and (b) hyperplane geometric calculations are heavy in regards to complexity. See figure below for pictorial representation of a **2 dimensional example** of the classification space:

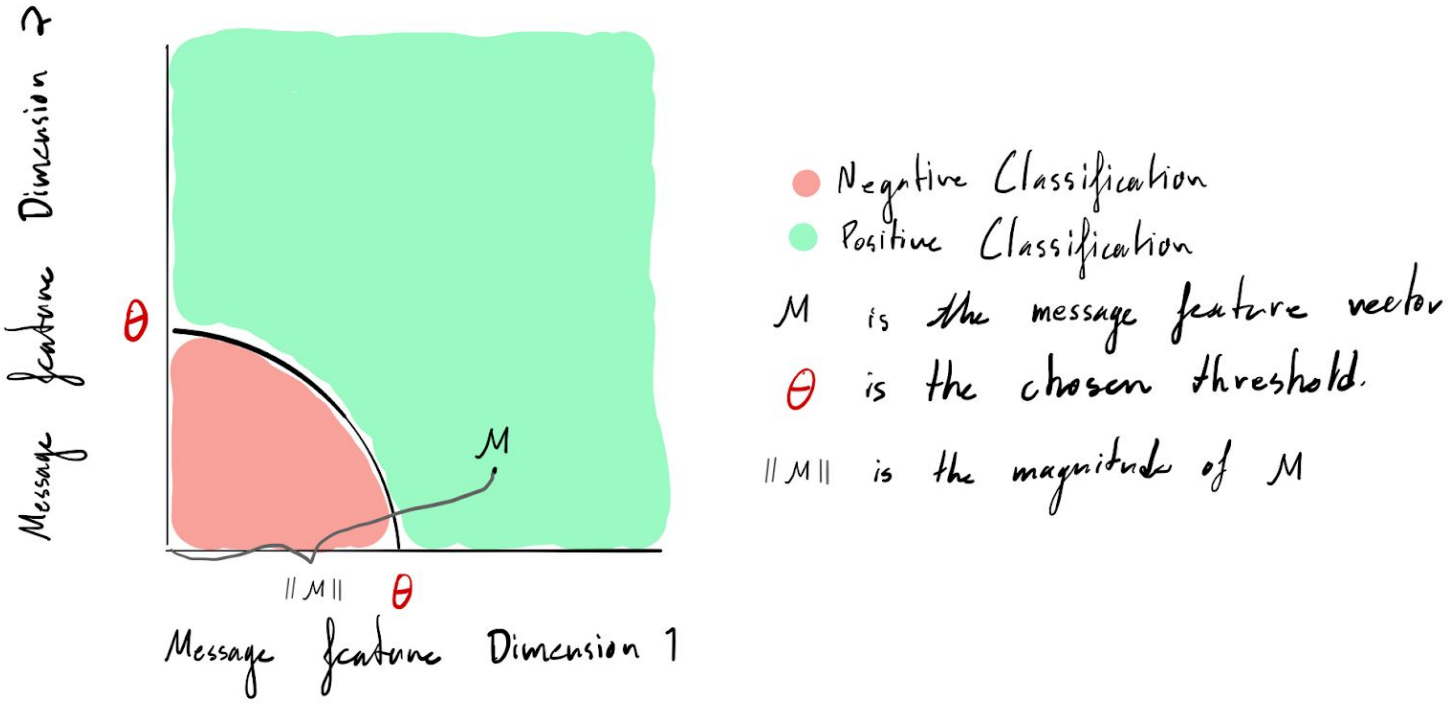

Figure: A two dimensional example of how the classifier makes a decision for a given message M, represented as a feature vector here in two dimensional space. Note this figure is analogous although it is a reduction in complexity since our model classifies in five dimensional space. The magnitude of M is compared to the threshold value theta, and a bool is returned as output from the classification.

False Positive and False Negative Reincorporation Learning

It may be possible to incorporate a learning component into this classification algorithm that considers past errors. Our classifier will keep a database of messages that users claim were errors on the classifier's behalf (whether that be false positives or negatives). Ideally the classifiers performance is improved as the event suggestion bot is subsequently used within its group chat, assuming the users give true errors claims as feedback.

I have found a previously made − highly prestigious − semantic comparison library at [3] which, given a database of sentences and a new message, computes a numerical quantity that denotes the message's semantic similarity to those previously seen. I will implement this into our model to compare new messages to old one's given as feedback. This incorporation will occur in feature dimensions 4 and 5 as previously described. Consider the following examples:

Example 1

Semantic meanings of sentences can be equivalent even when they have entirely unique words.

Message 1, an event suggestion:
"On August 1st, I would like our team to have a social."

Consider a subsequent message:
"The next long weekend my employees should meet for a break from work."
This message shares no words in common with the previous, yet they are conveying the same meaning. How can we label both as positive instances of event suggestion (see [3])?

Example 2

Semantic meanings of sentences not only depend on its constituent words but also the order in which they are permuted.

Message 1, an negative event suggestion:
"We should not meet today, since it is a holiday."
Suppose our classifier returns positive for this message since it contains a keyword "meet". Clearly this is a false positive. Hence we add this instance to our library of false positives.

Consider a subsequent message:
"Today is a holiday, we are not working or meeting."
Although this message contains a keyword "today", this is a negative instance. Using our sentence semantic comparator, it would score high in comparison to the previously known false positive, hence our classifier may improve its performance on this message.

RESULTS

Manual Testing without Feedback Learning

Thirty messages gathered from [4] and self written messages that we believed the bot should classify as a positive instance were labelled and put into “test.csv”. A script was written to iterate over parameter combinations for optimization. Dimensions one to three were iterated from 0.5 to 3.5 with an increment by 0.2, and with each permutation the model classified all 30 test messages with its performance simultaneously measured. Theta, the threshold value was set to 3. The top scoring parameters were: M_c = [0.8, 4.3, 0.5, 0, 0] scoring 27/30, **90%** classification success rate,

Manual Testing with Feedback Learning

The same thirty messages were fed to train the genism Doc2Vec model explained at [5]. After testing the trained model on the same instances, the result was 100% classification success rate, which was expected.

CONCLUSION

The outlined classifier was implemented for Google’s Software Product Sprint program in the 2 weeks allotted for our project. The reduced timeline meant our team was looking for a quick solution that could perform relatively well on manually created test data − we believed to have accomplished this. The algorithm outlined here is to be run as a bot within any group messaging system. The bot is then able to automate event planning through constantly ‘listening’ to its users messages sent in the chat. Each message sent is given to this classification algorithm, where it extracts and quantifies the relevant features to populate its five dimensional feature vector. Once a message is quantified, it is scaled (weighted) by predetermined constants. The magnitude of the scaled feature vector taken and compared to a threshold for binary classification.

Our team hopes this solution of automating event scheduling can increase productivity among working teams where scheduling meetings for a large team can be tedious.

BIBLIOGRAPHY

1. W. K. Nicholson, *Linear Algebra with Applications*. Open Textbook Library, 2018.
2. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition*. Springer New York, 2009.
3. RaRe-Technologies, "RaRe-Technologies/gensim," *GitHub*. [Online]. Available: https://github.com/RaRe-Technologies/gensim.
4. F. Eight, "Twitter US Airline Sentiment," *Kaggle*, 16-Oct-2019. [Online]. Available: https://www.kaggle.com/crowdflower/twitter-airline-sentiment/data#.
5. "gensim: topic modelling for humans," *Radim: Machine learning consulting*. [Online]. Available: https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html#sphx-glr-auto-examples-tutorials-run-doc2vec-lee-py.

INSTALLATION OF MODEL

For this model to work, you need to install:

pandas
sciPy
numPy
nltk
gensim

And then its tokenizer, from the python console:

```
>>> import nltk
>>> nltk.download('punkt')
```