

Subclassification of Intent using Naive Bayes.

Lalit Pathak
Student, VESIT.
Dept. of Computer
Engineering.
D17B , 53.

Akash Patil
Student, VESIT.
Dept. of Computer
Engineering.
D17B , 54.

Hardik Patil
Student, VESIT.
Dept. of Computer
Engineering.
D17B , 55.

Problem statement

The proposed system aims at automatically identifying and classifying dynamically, socially produced online data into intent categories and proactively mine information which is directly or indirectly linked to business opportunities. The outcome of this project will empower marketers and business owners to derive knowledge from large amounts of online data and to formulate decisions, converting data into actionable knowledge. It will also help customers to know more about the product or service that they intend to buy, thereby making it easier for them to select a low-cost and better quality product.

The Framework is designed for :

- 1) Classify the reviews into four categories- Purchase, Recommendation, Suggestion, wish.
- 2) The framework is able to visualize the results via interactive and intuitive charts and graphs for both managers and customers.
- 3) It can provide suggestion to the users about particular product or service.
- 4) It can suggest business intellectuals for any changes or recommendations in existing product and service.
- 5) Analyses the purchase intents and displays the information in dashboard.

Introduction

We have collected the dataset from Tweepy. We have create a web application comprising of three functionalities viz. Analysis visualization using pie charts, to train the model and check the accuracy and to predict the class of review.

Dataset

We have collected the real-time dataset from tweepy and used that dataset for subclassifying rev

Algorithm

Multinomial Naive Bayes:

1)We use NLTK (natural language toolkit) for 2 things:

- breaking up sentences into words (**tokenization**): “have a nice day” tokenizes to a list of individual words: “have”, “a”, “nice”, “day”
- reducing words to their stem (**stemming**): “have” stems to “hav” which allows it to be matched with “having” (same stem). There are various stemmers that we can use, for today we’ll use the Lancaster stemmer.

2)Next step is to provide some training data.

A few sentences that are associated with each intent (a “class”). If the user says “good day”, we want that to be the “greeting” intent.

3)Next we organized our data into 2 dictionaries:

You’ll notice that this is a list [] or dictionaries {}. Each dictionary has attributes: “class” and “sentence”. You could easily add additional attributes such as “responses” which could be what the chat-bot responds with when the intent is classified.

We’ve now organized our data into 2 dictionaries: **corpus_words** (each stemmed word and the # of occurrences) **class_words** (each class and the list of stemmed words within it). Our algorithm will use these data structures to do its thing. Notice the “corpus” (an NLP term) a collection of all stemmed words. The stem of “make” is “mak” so that it matches the stems for “making”, as a simple example. Notice each class generates a total score for the # of words that match. Study the above example closely and make sure it makes sense before proceeding. We can significantly improve our algorithm by accounting for the *commonality* of each word. The word “is” should carry a lower weigh than the word “sandwich” in most cases, because it is more common.

4)And now we can test our classifier with a few example inputs.

5) Our next step is to code our algorithm, our first version will treat each word with equal weight. In next version, for each input, class score is calculated and the class having high score is assigned to input.

Some advantages of decision trees are:

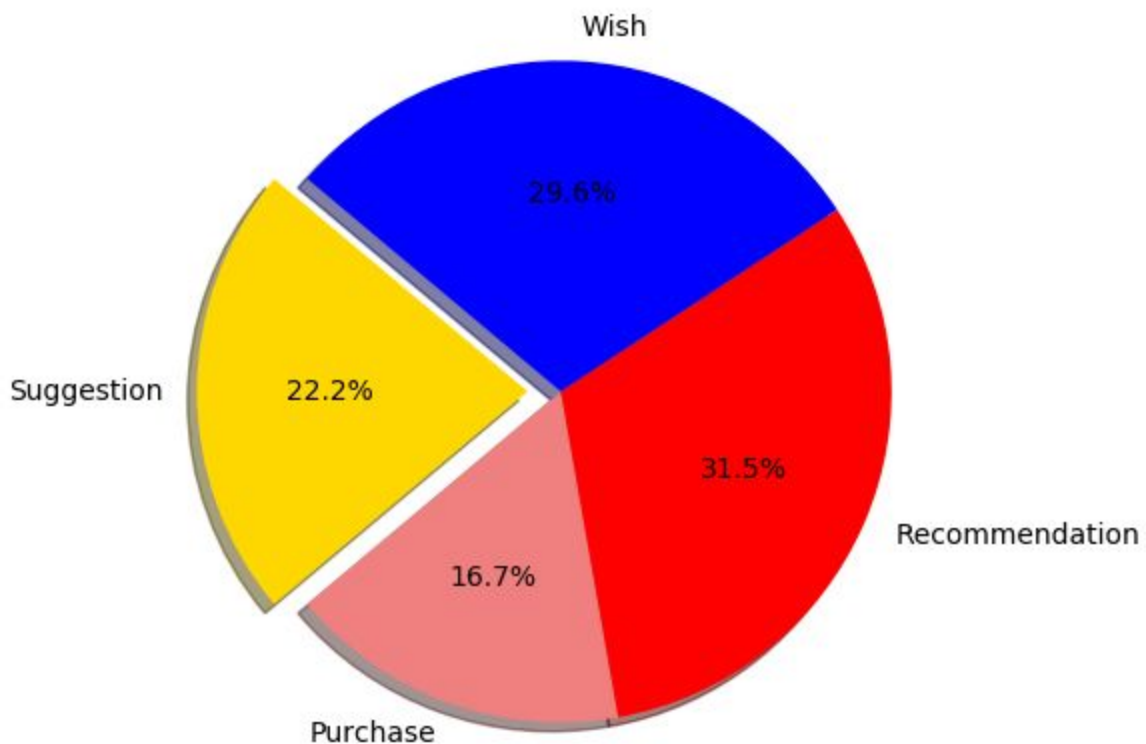
- Very simple, easy to implement and fast.
- If the NB conditional independence assumption holds, then it will converge quicker than discriminative models like logistic regression.
- Even if the NB assumption doesn't hold, it works great in practice.
- Need less training data.
- Highly scalable. It scales linearly with the number of predictors and data points.
- Can be used for both binary and multi-class classification problems.
- Can make probabilistic predictions.
- Handles continuous and discrete data.
- Not sensitive to irrelevant features.

The disadvantages of decision trees include:

- **Information theoretically infeasible** It turns out that specifying a prior is extremely difficult. Roughly speaking, we must specify a real number for every setting of the world model parameters. Many people well-versed in Bayesian learning don't notice this difficulty for two reasons:
 - They know languages allowing more compact specification of priors. Acquiring this knowledge takes some significant effort.
 - They lie. They don't specify their actual prior, but rather one which is convenient. (This shouldn't be taken too badly, because it often works.)
- **Computationally infeasible** Let's suppose I could accurately specify a prior over every air molecule in a room. Even then, computing a posterior may be extremely difficult. This difficulty implies that computational approximation is required.
- **Unautomatic** The "think harder" part of the Bayesian research program is (in some sense) a "Bayesian employment" act. It guarantees that as long as new

learning problems exist, there will be a need for Bayesian engineers to solve them.

Analysis



The project is a classification based model. The Naive Bayes gives an accuracy between 83-90%. The reason for varied accuracy for the same dataset is that the model divides the dataset into training and testing parts. 80% of the dataset is used for training and 20% for testing. So each time the model chooses different tuples for training and testing and it is not kept constant. In a way, each time a new training and testing dataset is generated even with the same csv file being used.

This can be observed in the GUI output.

Software Specification

1. Python 3.6
2. Flask


Tools Used

Anaconda- Jupyter notebook.

GUI Screenshots

Our Exclusive Services


Envision Industry is product to guide the decision-making process for Business Intelligence.



ANALYSIS OF CUSTOMER SENTIMENTS

The Path to Success


[view](#)



MARKETING STRATEGY ADVISING

Expert Guidance

[view](#)

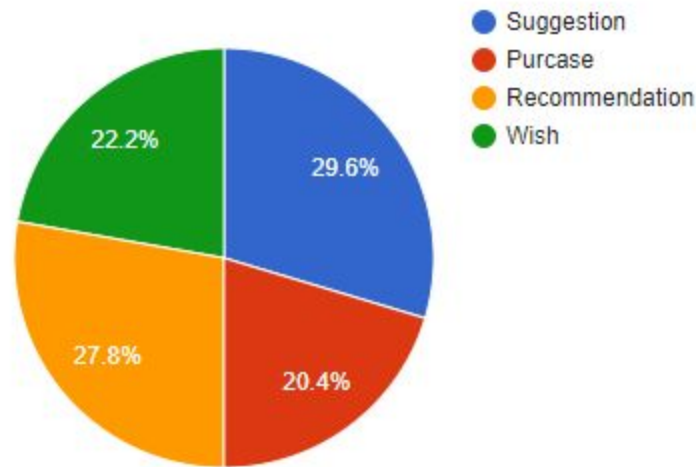


CUSTOMER'S INTENT ANALYSIS

A Comprehensive Approach

[view](#)

My Average Day



Conclusion

Thus we have trained our model for Subclassification of Intents using naive bayes. The application is developed in python using the Flask framework. We managed to achieve an accuracy in the range 83-90%.

References

- [1] <https://www.kaggle.com/datasets>
- [2] <https://chatbotslife.com/text-classification-using-algorithms-e4d50dcba45>
- [3] <http://cl.indiana.edu/~md7/16/555/slides/13-nltk2/13-nltk2-2x3.pdf>
- [4] <https://chatbotslife.com/text-classification-using-algorithms-e4d50dcba45>
- [5] https://en.wikipedia.org/wiki/Decision_tree_learning
- [6] <http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/>
- [7] https://www.researchgate.net/publication/289009275_Prediction_of_diabetes_using_decision_trees
- [8] http://bmjopen.bmj.com/content/6/12/e013336?utm_source=trendmd&utm_medium=cpc&utm_campaign=bmjopen&trendmd-shared=1&utm_content=Journalcontent&utm_term=TrendMDPhase4
- [9] <https://www.sciencedaily.com/releases/2016/06/160623115738.htm>

