Baby steps in a short-text classification with python

My personal horror story

Alisa Dammer
me: alisadammer.com
@FedorinoGore_90

July 12, 2017
Structure

Initial information collection

Award winning model

Going live

Did I learn anything?

Questions?
What can I do with a text

- Part of the speech tagging
- syntax model
- classification
- text generation
- translation

Binary classification it is!
We are a great company working in the health care sector. We are searching for a secretary for our chief doctor. We want you to work with papers answer calls, make coffee. The salary is good!
KLDB vs ISCO

43412
Informatics, Software development, Assistant/low level complexity

43494
Informatics, Software development, CTO, Tech Lead
Basic tools

- nltk
- sci-kit
- gensim
Evaluation tools

The confusion matrix is a table that is often used to describe the performance of a classification model (or in other contexts, a statistical model). The table is composed of two dimensions: actual and predicted, with four possible outcomes:

- True Positive (tp): The model correctly predicts the positive class when it is actually positive.
- False Positive (fp): The model incorrectly predicts the positive class when it is actually negative.
- True Negative (tn): The model correctly predicts the negative class when it is actually negative.
- False Negative (fn): The model incorrectly predicts the negative class when it is actually positive.

The confusion matrix is a useful tool for evaluating the performance of a classification model, as it provides a summary of the model's accuracy, precision, recall, and other metrics.
Let the evaluation begin!

- Bernoulli classification
- Naive Bayesian
- Support Vector Machine
- Decision Tree
Tuning up

- Tweak data set as a whole
- Tweak each item in the data set
Tweaking the item

- Add information
- Remove information
- Stemm the crap out of it
Data transformed!
import nltk.NaiveBayesClassifier as nbc

def build_nb(train):
    modelTrained = nbc.train(train)
    return modelTrained

def train_nb():
    sample = load("path/filename")
    train, test = splitSample(sample, 0.7)
    train = formatForNLTK(train, True, lang)
    test = formatForNLTK(test, True, lang)
    model = build_nb(train)
    getEstimationResults(model, test, labels)
    savePickle("models/classify.pkl", model)
Every day we’re modelling

Time required to train NB is 0.6297673170047347
General TP is 224
General FP is 119
overall accuracy is 0.6530612244897959
confusion matrix is
[[ 53  32  0]
 [ 16 112  0]
 [  0  0  0]]
Dooooooom!

Good news everyone!
We're all doomed!
Reconnection

- Jython
- Starting python scripts inside of the java code
- Rewrite in Java
- Message brokers
- REST
... model = readPickle("model.pkl")
@app.route('/classify', methods=['POST'])
def classify():
    formatted = {}
    results = {}
    if request.method == "POST":
        item, lang = validate(request)
        if lang != expected:
            error_response(lang, model)
        else:
            formatted[model.label] = [item]
            classify(results, formatted, lang, model, model.label)
    logging.info("Classified!")
    return jsonify(results)
Is the problem solved?

- Spend more time on base research
- Don’t go too deep
- Try graphs first
- Don’t be afraid to change the data itself
- Monitoring over historical data
- Have a minimal quality test
- Cross validation is a thing
Thanks for the patience!
Maybe useful information

Tutorials:
▶ https://pythonprogramming.net/naive-bayes-classifier-nltk-tutorial/
▶ http://www.nltk.org/book/ch06.html
▶ http://scikit-learn.org/stable/tutorial/text-analytics/working_with_text_data.html
▶ http://www.nltk.org/_modules/nltk/metrics/confusionmatrix.html

Basic:
▶ http://www.linguistics.fi/julkaisut/SKY2006_1/1.6.6.%20NIVRE.pdf
▶ https://rstudio-pubs-static.s3.amazonaws.com/79360_850b2a69980c4488b1db95987a24867a.html
▶ https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words

Deep:
▶ http://karpathy.github.io/neuralnets/
▶ http://course.fast.ai/lessons/lesson2.html