# **Notes for Objectives Section**

## Main Points from Lectures and Readings for Sentiment Analysis for this Midterm (Week 10)

### Applications of Sentiment Analysis

* Applications of sentiment analysis: movie/product reviews, public opinion (politics, **consumer confidence**), prediction (stocks/markets).
* Understanding the “voice of the customer”, how products/services are perceived in the marketplace.
* For authors = understanding how your book has been received.
* Voice of the customer is very important in Amazon/e-commerce (sentiment expressed in social media etc., a whole industry is based on measuring that sentiment and reporting it back to companies).
* **Many products and services are discussed in soc media/online**, lots of work done on reporting sentiment back to organizations to inform their decision-making!
* **Recommendation of movies/books based on comments on social media**.
* Consumer confidence
* Sentiment about government initiatives and other socioeconomic phenomena.
* Predicting stock movies/box-office movie sales.
* **Clear business value.**

### Affective States

* Scherer typology of affective states: personality traits (long period of time), mood, emotion, interpersonal stances, **attitudes (the focus)**.
* Attitudes = enduring, affectively colored beliefs, dispositions towards **objects** or **persons** (liking, hating, loving, valuing). *Most interested in these phenomena in sentiment analysis techniques*
* **Sentiment analysis is the detection of attitudes.**
* Is the attitude of a given text **positive** or **negative**?

### NB Approach

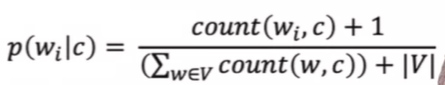
* Estimate priors of each **class** 🡪 ignore unknown words 🡪 estimate **likelihoods** based on the terms in the test data 🡪 compare scores per class.
* We tend to **ignore unknown words** in sentiment analysis, unlike in other text classification tasks, where we deal with this in other ways.
* **Estimating priors for each class:**
  + P(positive) = *number of positive samples / total number of samples*
  + Do the same for other classes.
* **Ignoring terms (e.g. “was”) that do not appear in the training data but are in the test-dataset is very important for sentiment analysis!**
* **Estimating likelihoods for Naïve Bayes theorem (the likelihood of observing a word given the sentiment class):**
  + **Simple counting technique** used in language modelling
  + For each word/token in the test-sentence (except for the ignored, unknown words), do this step first to calculate the likelihood:

***Probability(some word | sentiment class)*** *=*

*COUNT(number of times the word appears in this particular class of documents) + 1*

*/*

*(Number of tokens or words, called “countable terms” in this particular sentiment class +* ***Total*** *size of vocabulary (unique terms) for all the classes, which is called the “normalizing constant”)*



* Laplace smoothing is used to address the problem of 0-counts.
* Then, to calculate the score/posterior probability of the sentence/document belonging to a sentiment class, we multiply all the likelihoods (one-per-token/word) calculated in the step above for each word by the prior for that sentiment class…
* We then compare scores to find the most likely class.
* **Remember:** unknown words are ignored in sentiment analysis using Naïve-Bayes.

### Optimizations that are Important in Sentiment Analysis

* **Ignoring Frequency Counts:** Usually in text categorization, we account for **term frequency** and **inverse document frequency (TF-IDF)**! (Look this up in more detail)
* For sentiment, we can throw away the TF-IDF **frequency information**.
* This adds little value to the analysis.
* Occurrence matters more than frequency! Whether something occurs is more important than how frequently something occurs.
* **Binary** multinomial NB: we limit the word counts to 1 or 0.
* We do a **binary count**, we do not record the frequencies of how often we saw the word ‘is’, but we just do a 1 if it exists in the sentence and 0 otherwise.
* **The Problem of Negation and Sentiment Polarity:**
  + Negation changes polarity, i.e. “the book was good” vs “the book was not good”, changes the polarity of the **entire utterance**! Another example: “it’s not bad”
  + Therefore, taking *not* as a feature on its own is not very effective.
  + **Simple baseline method:** add NOT\_ to every word between the **negation** and the **following punctuation**, e.g.
    - “I didn’t like this book, but…” becomes “I didn’t NOT\_like NOT\_this NOT\_book, but…”
    - **How to actually do this in Python?**
  + Then, *these* are the **new features** you would use to classify the likelihoods, or some other classifier technique.
  + Must modify the feature set!

### Sentiment Lexicons

* Use when training data is limited
* Don’t need training data
* Manually curated wordlists
* E.g. MPQA subjectivity lexicon, General Inquirer (for polarity detection, but has more nuances)
* **To use sentiment lexicons in text classification for sentiment polarity detection:**
  + Add a **feature** for whether a word in the test dataset is a member of the positive class, or whether it is a member of the **negative** class
  + You basically keep a running count (for each document) of all the positive and all the negative words: e.g. *count\_pos(wi) if wi is in the set of positive words, count\_neg(wi) if wi is in the negative class*.
  + Can improve basic sentiment analysis, an important optimization.

### Practical Tips in Jupyter Notebook for Sentiment Analysis (nltk Movie Reviews)

* Create a freq distribution of **only the most COMMON words** (all lowercase? Are you sure you want to do this?), and only take the top-n words… this limit means it will be fast.
* **Feature Extraction:**
  + Write helper functions for these.
  + For each document, apply a helper function that converts the document to a **set** (faster than a list), and for each of the top-most common words, we create a dict-pair, which is the word as the key and True/False as the value, to indicate if the word is present in the document.
  + Can do n=500, etc.
  + **Look at the whole size of the vocabulary set!**
* **How to do feature sets for each document (poem segment):**
* Use the nltk NaiveBayesClassifier: NBclassifier.show\_most\_informative\_features(5)
* DTclassifier, MaxEntclassifier (many iterations), SVM, LogisticRegression
* Decide what to do with stopwords.