Principles of Sentiment and Emotion Mining

Why compute affective meaning?

Detecting:

- sentiment towards politicians, products, countries, ideas
- frustration of callers to a help line
- stress in drivers or pilots
- depression and other medical conditions
- confusion in students talking to e-tutors
- emotions in novels (e.g., for studying groups that are feared over time)

Could we generate:

- emotions or moods for literacy tutors in the children's storybook domain
- emotions or moods for computer games
- personalities for dialogue systems to match the user

Connotation in the lexicon

Words have connotation as well as sense

Can we build lexical resources that represent these connotations?

And use them in these computational tasks?

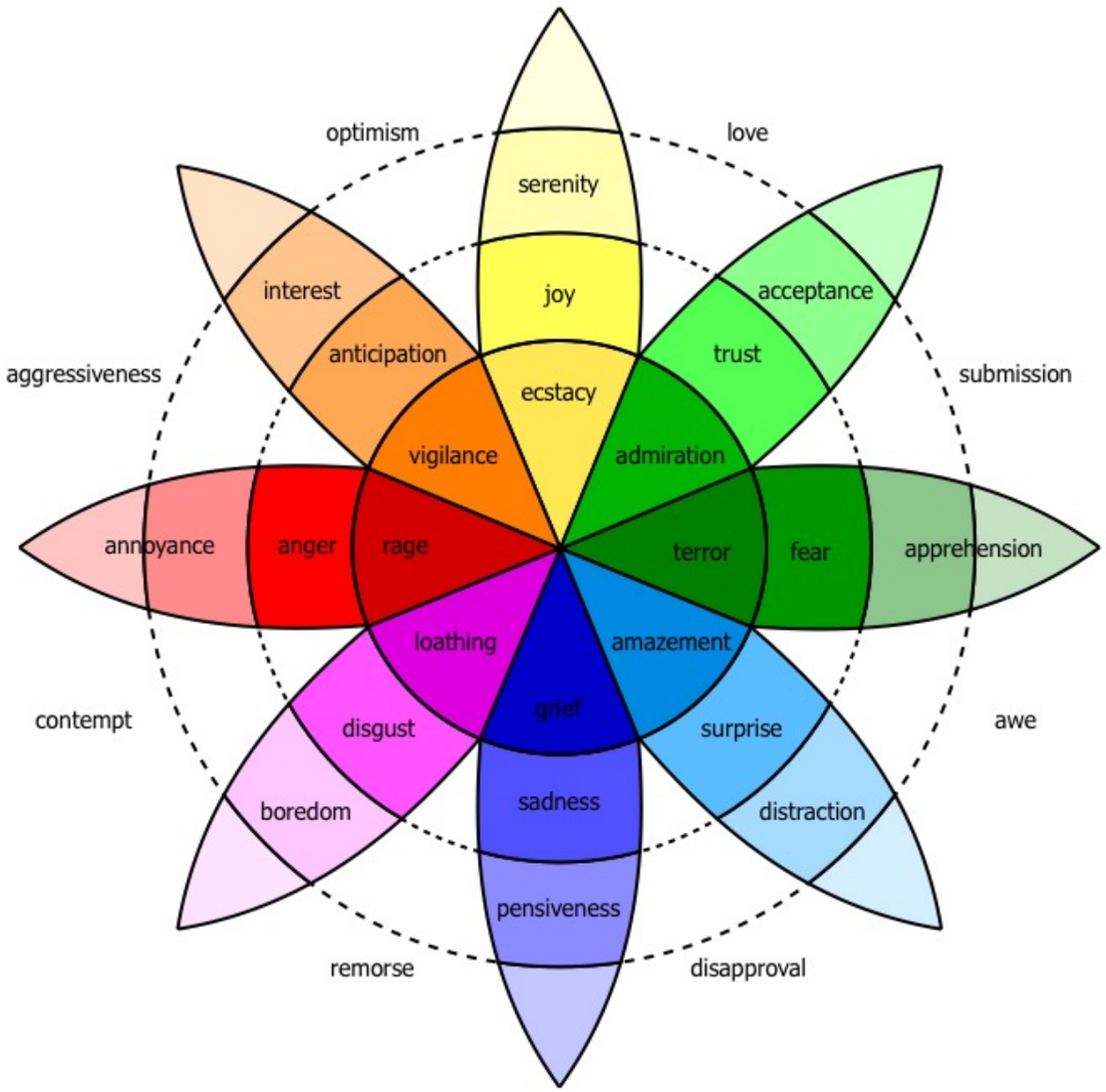
Ekman's 6 basic emotions:

Surprise, happiness, anger, fear, disgust, sadness

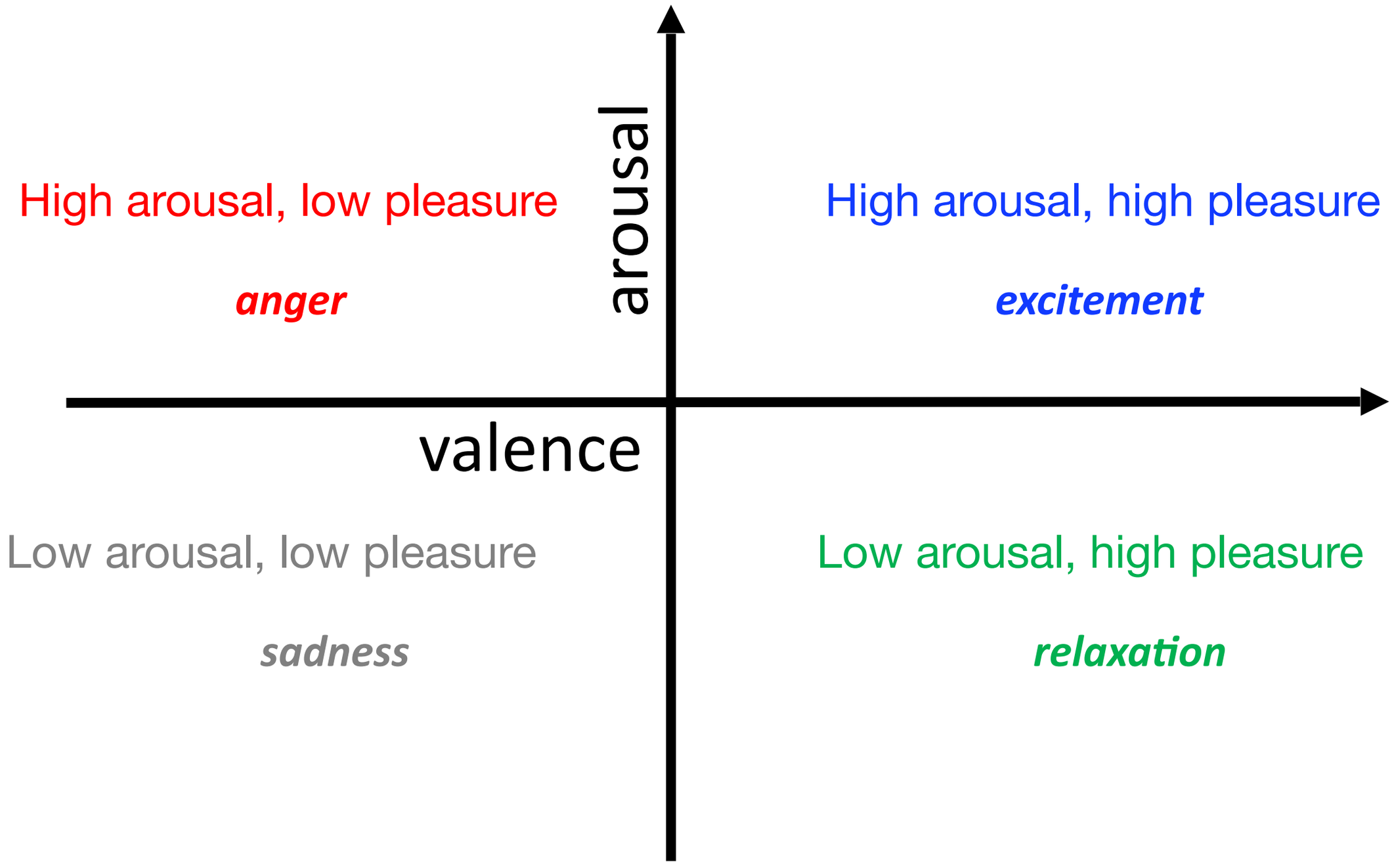
Ekman & Matsumoto 1989

Plutchick's wheel of emotion

- 8 basic emotions
- in four opposing pairs:
  - joy–sadness
  - anger–fear
  - trust–disgust
  - anticipation–surprise



Valence/Arousal Dimensions



Words with consistent sentiment across lexicons

Positive	admire, amazing, assure, celebration, charm, eager, enthusiastic, excellent, fancy, fantastic, frolic, graceful, happy, joy, luck, majesty, mercy, nice, patience, perfect, proud, rejoice, relief, respect, satisfactorily, sensational, super, terrific, thank, vivid, wise, wonderful, zest
Negative	abominable, anger, anxious, bad, catastrophe, cheap, complaint, condescending, deceit, defective, disappointment, embarrass, fake, fear, filthy, fool, guilt, hate, idiot, inflict, lazy, miserable, mourn, nervous, objection, pest, plot, reject, scream, silly, terrible, unfriendly, vile, wicked

NRC Word-Emotion Association Lexicon

Mohammad and Turney 2011

```
amazingly    anger    0
amazingly    anticipation 0
amazingly    disgust 0
amazingly    fear     0
amazingly    joy      1
amazingly    sadness 0
amazingly    surprise    1
amazingly    trust      0
amazingly    negative    0
amazingly    positive    1
```

Emotion Mining Tools

- Example: text2emotion library for Python

  https://pypi.org/project/text2emotion/

  Note when installing it, please install emoj 1.6.3 first (the last one is incompatible with text2emotion)

```
pip install emoji~=1.6.3
```

Text2emotion examples

```
import text2emotion as te

sentences=[
    "Today is a wonderful day!",
    "That night was so horrible.",
    "I can't understand what to do now.",
    "I didn't expect that",
    "I just discovered that"
]

for text in sentences:
    em=te.get_emotion(text)
    print(text,em)
```

Results

- Today is a woderful day! {'Happy': 1.0, 'Angry': 0.0, 'Surprise': 0.0, 'Sad': 0.0, 'Fear': 0.0}
- That night was so horrible. {'Happy': 0.0, 'Angry': 0.0, 'Surprise': 0.0, 'Sad': 0.0, 'Fear': 1.0}
- I can't understand what to do now. {'Happy': 0.0, 'Angry': 0.0, 'Surprise': 0.0, 'Sad': 1.0, 'Fear': 0.0}
- I didn't expect that {'Happy': 0.0, 'Angry': 1.0, 'Surprise': 0.0, 'Sad': 0.0, 'Fear': 0.0}
- I just discovered that {'Happy': 0.0, 'Angry': 0.0, 'Surprise': 1.0, 'Sad': 0.0, 'Fear': 0.0}

Another example

```
sentences=["The cat is on the table",
           "The cat is on the table :-)",
           "The cat is on the table (happy)"]

for text in sentences:
    em=te.get_emotion(text)
    print(text,em)
```

What do you expect?

Result

The cat is on the table {'Happy': 0, 'Angry': 0, 'Surprise': 0, 'Sad': 0, 'Fear': 0}

The cat is on the table :-) {'Happy': 0, 'Angry': 0, 'Surprise': 0, 'Sad': 0, 'Fear': 0}

The cat is on the table (happy) {'Happy': 1.0, 'Angry': 0.0, 'Surprise': 0.0, 'Sad': 0.0, 'Fear': 0.0}

Lessons learned: for some tools it might be useful to preprocess emoji

Example Dataset

https://www.kaggle.com/c/sa-emotions/data

Sentiment Analysis

Sentiment Analysis

- Narrower than emotion mining
- The main goal is to determine whether a word or a sentence conveys a positive or negative mood
- We refer to it as “polarity”
- A sentence or a word could have a positive, neutral, or negative polarity

Sentiment Analysis: Applications

- Product review

- Useful to understand (automatically) whether a customer liked a product or not

Sentiment Analysis: Approaches

- Dictionary based: each word conveys a sentiment, positive, negative, or neutral
    - The sentence sentiment is computed by aggregating the sentiment of single words
- Natural Language Parsing: the tool interprets the whole sentence and provides an overall sentiment
- Machine Learning: the sentiment model is learned from existing datasets

Sentiment Analysis with NLTK

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')

sa= SentimentIntensityAnalyzer()

sentence="I'm very disappointed"

print(sa.polarity_scores(sentence))
```

Examples of output

- I'm very disappointed

  {'neg': 0.629, 'neu': 0.371, 'pos': 0.0, 'compound': -0.5256}

- I’m not very disappointed

  {'neg': 0.0, 'neu': 0.52, 'pos': 0.48, 'compound': 0.4158}

Note how it is able to handle the negation!

Examples of output

- I’m happy

  {'neg': 0.0, 'neu': 0.213, 'pos': 0.787, 'compound': 0.5719}

- I’m happy!!

  {'neg': 0.0, 'neu': 0.189, 'pos': 0.811, 'compound': 0.6467}

- I’m happy :-)

  {'neg': 0.0, 'neu': 0.143, 'pos': 0.857, 'compound': 0.7184}

Learning Sentiment

- We can learn sentiment from existing datasets and create a sentiment classifier
- This is how many tools work
- Indeed this is a good project for the course

Learn sentiments: hints

- Just removing punctuation and special characters is a bad idea
- Before removing punctuation, you may want to transform meaningful combinations of characters into Words
- Examples:
  - replace “:-)” with “happy”
  - replace “:-(“ with “sad”
  - replace “!!” (or more) with bigExclamation
  - replace “?!?” with Doubtful
  - replace “!” with Exclamation (do this after the previous ones)
  - etc..
- More difficult:
  - Every time a word is upper case, this means one is shouting
  - E.g. if you have “WARNING” you may want to count this as double words, i.e., “warning” and “shoutwarning”
  - Note: the Python function `isupper()` tells whether a string is all uppercase

Very simple example

- Classifying the sentiment of movie reviews using Naive Bayes

- Dataset: IMDB reviews
  https://www.kaggle.com/columbine/imdb-dataset-sentiment-analysis-in-csv-format

- Note: In this example I'm not using any special preprocessing

- This is left for your projects

The code - preprocessing

```
import pandas as pd
from nltk.corpus import stopwords
import gensim
import numpy as np
from bs4 import BeautifulSoup

def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

def transformText(text):
    stops = set(stopwords.words("english"))
    # Remove HTML
    text = strip_html(text)
    # Convert text to lowercase
    text = text.lower()
    # Strip multiple whitespaces
    text = gensim.corpora.textcorpus.strip_multiple_whitespaces(text)
    # Removing all the stopwords
    filtered_words = [word for word in text.split() if word not in stops]
    # Preprocessed text after stop words removal
    text = " ".join(filtered_words)
    # Remove the punctuation
    text = gensim.parsing.preprocessing.strip_punctuation(text)
    # Strip all the numerics
    text = gensim.parsing.preprocessing.strip_numeric(text)
    # Removing all the words with less than 3 characters
    text = gensim.parsing.preprocessing.strip_short(text, minsize=3)
    # Strip multiple whitespaces
    text = gensim.corpora.textcorpus.strip_multiple_whitespaces(text)
    # Stemming
    return gensim.parsing.preprocessing.stem_text(text)
```

Discussion

- We are using the Beautiful Soup library to extract text from HTML
- Very simple in this case:

```
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()
```

- However, you and do more advanced things with the library, such as extracting text only from certain tags, etc.

Creating the model

```
dataset=pd.read_csv("IMDB Dataset.csv")
print(dataset.describe())
print(dataset['sentiment'].value_counts())

dataset['review'] = dataset['review'].map(transformText)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(dataset['review'], dataset['sentiment'],
                                                    test_size=0.33, random_state=10)
```

```
from sklearn.naive_bayes import MultinomialNB

#Build the counting corpus
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)

## Get the TF-IDF vector representation of the data
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

clf = MultinomialNB()
clf.fit(X_train_tfidf, y_train)

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB(alpha=1)
clf.fit(X_train_tfidf, y_train)
```

Discussion

- We load the dataset from a .csv file
- The review is in the “review” column
- The sentiment is in the “sentiment” column

Performing the prediction…

```
#indexing the test set
X_new_counts = count_vect.transform(X_test)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

#performing the actual prediction
predicted = clf.predict(X_new_tfidf)

#printing evaluation reports
from sklearn import metrics
print(metrics.confusion_matrix(y_test,predicted))

print(metrics.classification_report(y_test, predicted))
```

Output...

```
[[7302 1055]
 [1182 6961]]
```

	precision	recall	f1-score	support
negative	0.86	0.87	0.87	8357
positive	0.87	0.85	0.86	8143
accuracy			0.86	16500
macro avg	0.86	0.86	0.86	16500
weighted avg	0.86	0.86	0.86	16500

Exercises (for your project)

- Try to perform a better preprocessing
- Try to use SVM instead
- Compare with the results produced by the NLTK sentiment analysis tool
- Try to implement emotion mining using machine learning classifiers