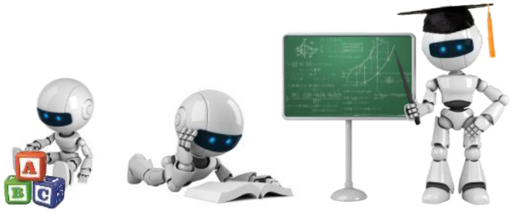
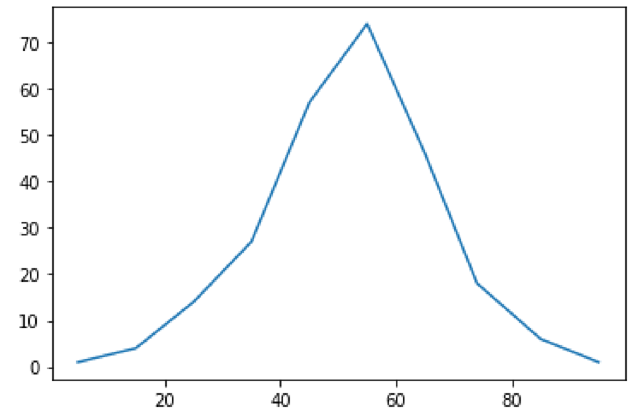
CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Logistical Notes

- Quizzes:
  - Quiz Period: 2:05-2:20 and still 10 minutes
  - Quiz Goal and Difficulty
  - We will have Quiz 2 next Tuesday



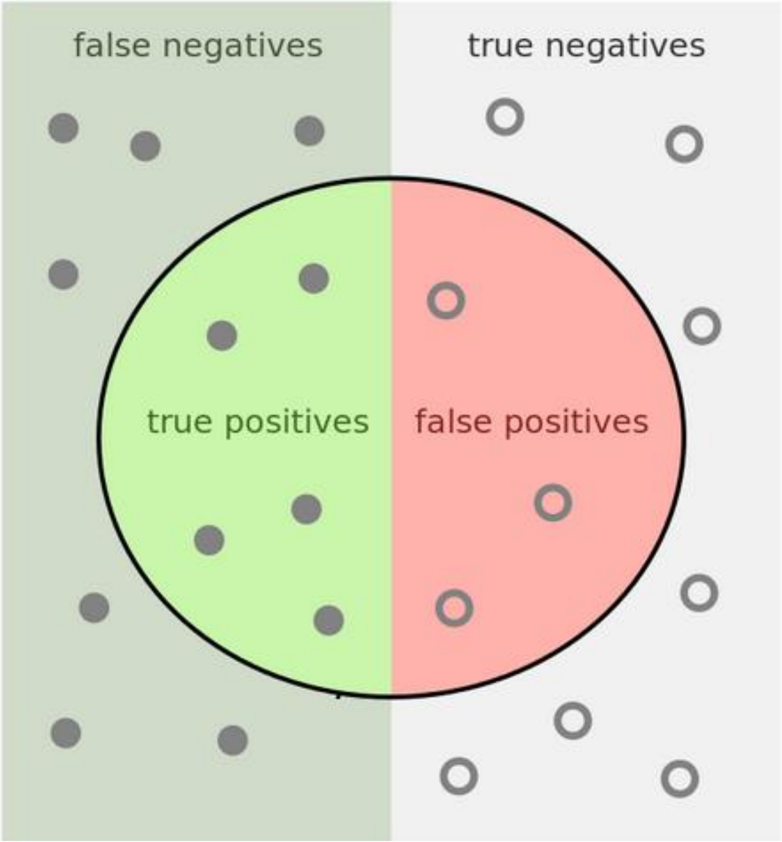
- Project:
  - 42 Groups < 52 groups
  - Groups of 6?
- HW1:
  - Please check the homework description for edits done based on feedback from the Slack channel and prepare your report accordingly

Model Evaluation Process

- We use a training dataset for model selection
- A **good** parametric model along with a **suitable** training algorithm guarantees training a model that works well on the training data
- We need to validate that trained models **generalize** well on unseen data instances
- We need a second testing dataset which is fully independent of the training dataset
- We randomly split the annotated dataset into testing and training splits (sometimes, a validation set is generated as well)

Evaluation Metrics

- **Accuracy**: proportion of correctly classified items
- Accuracy can be dominated by **true negatives** (items correctly classified as not in a class).
- Sensitive with respect to imbalance



- Precision: $\frac{\text{True Positives}}{\text{True Positives+False Positive}}$
- Also called positive predictive value
- Recall: $\frac{\text{True Positives}}{\text{True Positives+False negative}}$
- Also called sensitivity
- Precision and recall are not useful metrics when used in isolation?
- We want our model to have good performance with respect to both metrics
- Implemented in sklearn

Evaluation Metrics

- Why having one measure is helpful? Optimization!

$$\text{F1} = \frac{2\ \text{Precision Recall}}{\text{Precision} + \text{Recall}} = \frac{2}{1/\text{Precision} + 1/\text{Recall}}$$

- F1 is biased towards the lower of precision and recall:

- harmonic mean < geometric mean < arithmetic mean
- F1=0 when Precision=0 or Recall=0

- Generalized F score:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

- Multiclass case?

Natural Language Representation

- Language processing hierarchy levels:
- **Documents**
- Sentences
- Phrase
- Words
- Sparsity in the NLP training datasets: natural language has a very huge space:
- Ex: Average Wikipedia page size is 580 words and English has ~1M words, yet the actual number of possibilities is far more.
- We need **interpretable** representations or **embeddings** to represent natural language data for model training
- One-hot representation: two large (15M words) and meaningless

  Hotel: [0,0,0,0,1,0,0,0,0,0,0,…,0,0,0]

  Motel:[0,0,0,0,0,0,0,0,0,1,0,…,0,0,0]

Similarity of Vectors

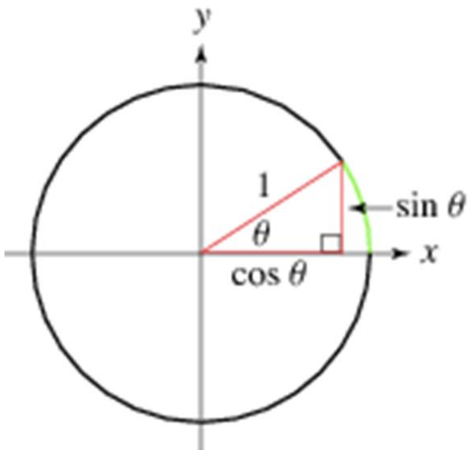
- Euclidean distance, i.e., geometric closeness :

\- Curse of dimensionality

- Dot product:

$a \bullet b = ||a||\,||b||\cos(\theta_{ab})$

$\quad = a_1b_1+a_2b_2+ \ldots +a_nb_n$



\- Cosine similarity (scale invariant)

$\cos\theta_{ab} = a \bullet b \;/\; ||a||\,||b||$ -> $1-\cos\theta_{ab}$ is a metric

\- Invariant with respect to the vector starting point

\- EX: Hotel: [0,0,0,0,1,0,0,0,0,0,0,…,0,0,0], Motel:[0,0,0,0,0,0,0,0,0,0,1,0,…,0,0,0]

Hotel'*Motel = 0

The Distributional Hypothesis

- Words that occur in the **same contexts** tend to have similar meanings (Zellig Harris, 1954)

- Example: nice, good

- Word relatedness association (Budanitsky and Hirst, 2006): related words **co-occur** in different contexts

- Example: cup, coffee

- If semantic similarity and association of words can be encoded into their representations, we may be able to address the challenge of sparsity

- In the absence of a particular word during training, we can rely on its synonyms that exist in the training dataset: Motel vs Hotel

- We can draw conclusions:

  Lecturers teach in the university-> Professors ___ in the university.

Vector Embedding of Words

- Represent words using dense vectors:
  - Latent Semantic Analysis/Indexing (SC Deerwester et al, 1988)
    - Term weighting-based model
    - Consider occurrences of terms at **document level**.
  - Word2Vec (Mikolov et al, 2013)
    - Prediction-based model.
    - Consider occurrences of terms at **context level**.
  - GloVe (Pennington et al, 2014)
    - Count-based model.
    - Consider occurrences of terms at **context level.**

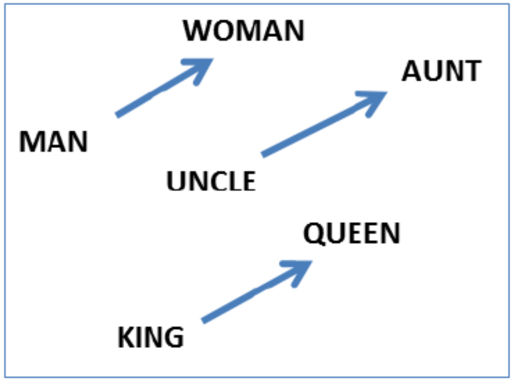
Word Embedding

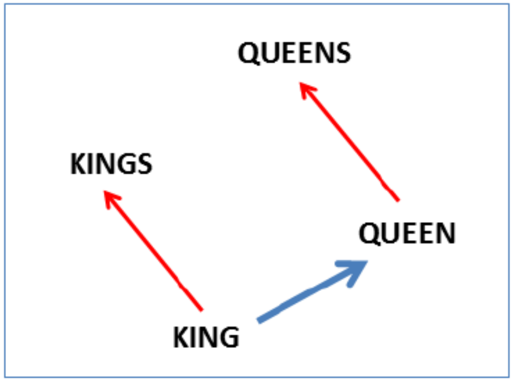
- Each word is represented by a vector:

- The same size is used for all words
- Relatively low dimensional (~300)
- Vectors for similar words are similar (measured in dot product)
- Vector operations can be used for

semantic and syntactic deductions, e.g.,

Queen – Woman + Man = King





- The key idea is to derive the embeddings from the distributions of word context as they appear in a large corpus.

Matrix Factorization

- We can form a matrix of M using the idea of Bag of Words: the word representations are highly sparse

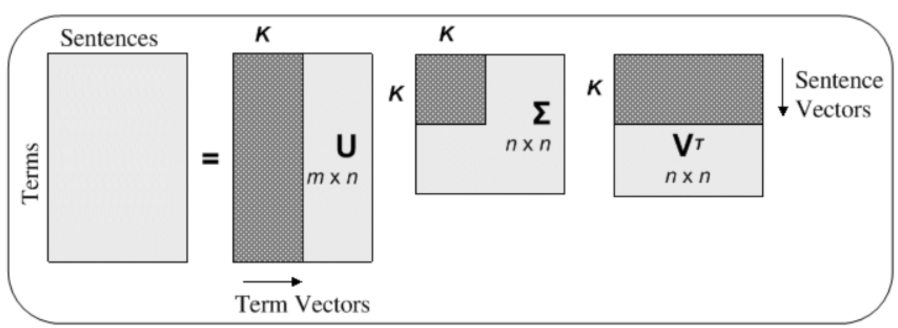
Words

Contexts

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

- Singular value decomposition (U, V are orthonormal)

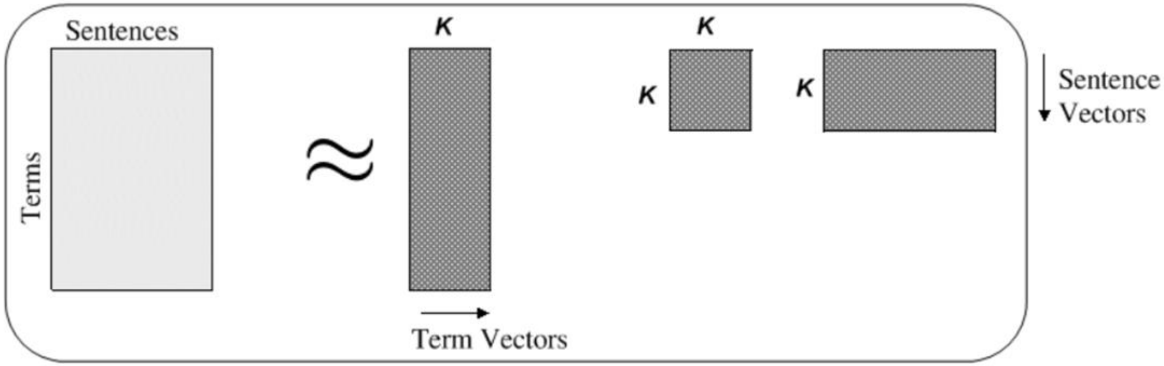
$$M_{m\times n} = U_{m\times n}\Sigma_{n\times n}V^T_{n\times n}$$



Matrix Factorization

- Many singular values are going to be zero or negligible

$$M_{m\times n} \approx U'_{m\times k}\Sigma'_{k\times k}V'^{T}_{k\times n}$$



$$m_l = u_l \Sigma' V'^T$$
$$u_l = m_l V' \Sigma'^{-1}$$

- We can use rows of U as word embeddings

- An old idea for dimensionality reduction (it is possible to use other matrix factorization methods, e.g., non-negative matrix factorization)
- Determining context is heuristic
- computational expensive with O(mn^2) cost for an n*m matrix
- Hard to incorporate new words

Word2Vec

- Core idea: find embeddings using a prediction task involving **neighboring words** in a huge real-world corpus.
- Input data can be sets of **successive word-patterns** from meaningful sentences in the corpus, e.g., "one of the most important".
- Try to build **synthetic** prediction tasks using these patterns, e.g., "(one of __ most important, the)"
- Train a model to solve the prediction task
- Embeddings are found as a **byproduct** of this process

- More specifically:
- **We consider a window with the center word w_t and "context words" $w_{t'}$ with a window fixed size, e.g., (t'=t-5, … t-1, t+1, … , t+5).**
- The model is assumed to be a two-layer neural network
- We train the network to predict all w_t given $w_{t'}$ such that $p(w_t|w_{t'})$ is maximized
- We learn embeddings such that the prediction loss is minimized, i.e., if two words occur in close proximity, their representations become similar.