NATURAL LANGUAGE PROCESSING

WITH DEEP LEARNING

University of Guilan

Lecturer: Javad PourMostafa

NLP981

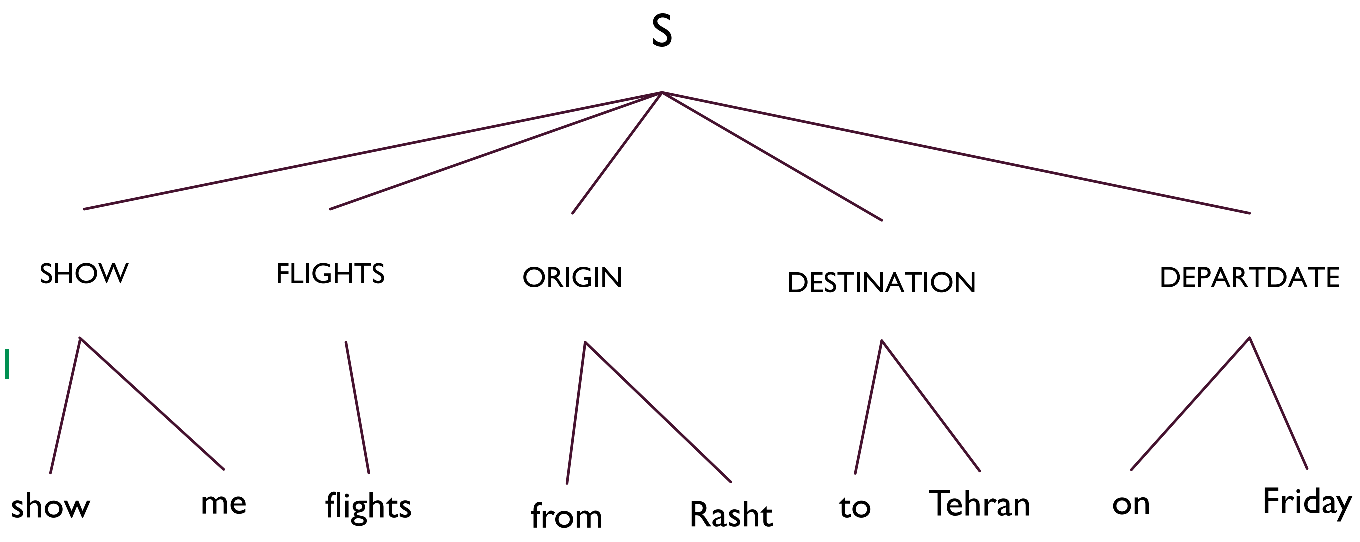
Introduction to NLP

MAIN APPROACHES

1. Rule-based Methods
   1. Regular Expression
   2. CFG
   3. …
2. Traditional ML algorithms and Probabilistic modeling
   1. Likelihood Maximization
   2. Supervised/Non-supervised algorithms
   3. …
3. Deep Learning
   1. RNN
   2. CNN
   3. ….

SEMANTIC SLOT FILLING: CFG

- What are Context Free Grammar?
  - A formal grammar
  - Every production rule is of the form V → w
  - V is a single nonterminal symbol and w is a string of terminals and/ or non-terminals
- Context Free Grammar:
  - SHOW → show me | i want | can i see | …
  - FLIGHTS → (a) flight | flights
  - ORIGIN → from CITY
  - DESTINATION → to CITY
  - CITY → Rasht | Tehran | Dubai | Isfahan | Istanbul



SEMANTIC SLOT FILLING: CFG (CONT.'S)

- Advantages vs. Disadvantages?
  - Usually done manually
  - Time Consuming
  - High Precision
  - Low Recall

CONFUSION MATRIX

- Machine Learning Fundamental
- Dividing data into Training set and Testing set
- Summarize performance on the Testing data
- Create confusion matrix

Predicted Values \ Actual Values	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

$$\text{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

SEMANTIC SLOT FILLING: CRF

- A Probabilistic Graphical Model
- Training Corpus:

ORIG DEST DATE

Show me flights from Rasht to Tehran on friday.

- Some Feature Engineering:
  - Is the word in a list of city names?
  - What is the previous slot?
  - Is the word capitalized?
  - ….

SEMANTIC SLOT FILLING: CRF (CONT'D)

- Define your Conditional Random Field model:

P(tags | words) = …

Features

Parameters Θ

- Training:

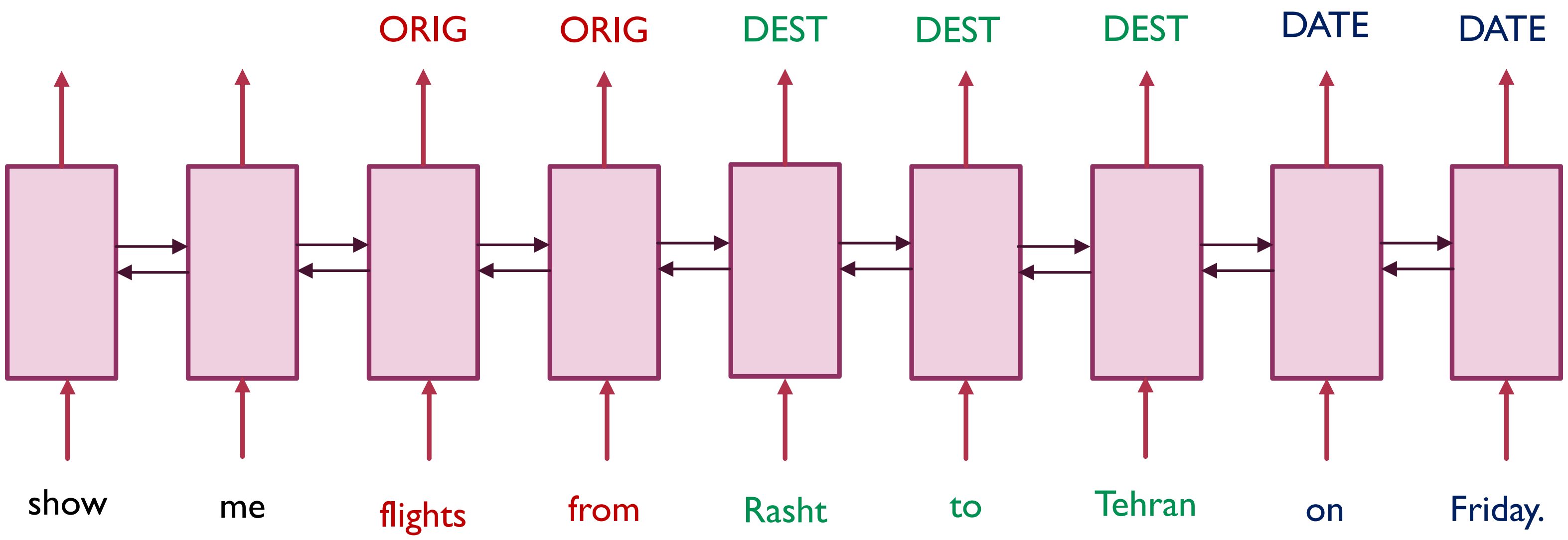
$$\text{P(tags | words)} \longrightarrow \text{Max by } \Theta$$

- Inference/Evaluation/Test:

$$tags^* = \text{argmax P(tags | words)}$$

SEMANTIC SLOT FILLING: LSTM

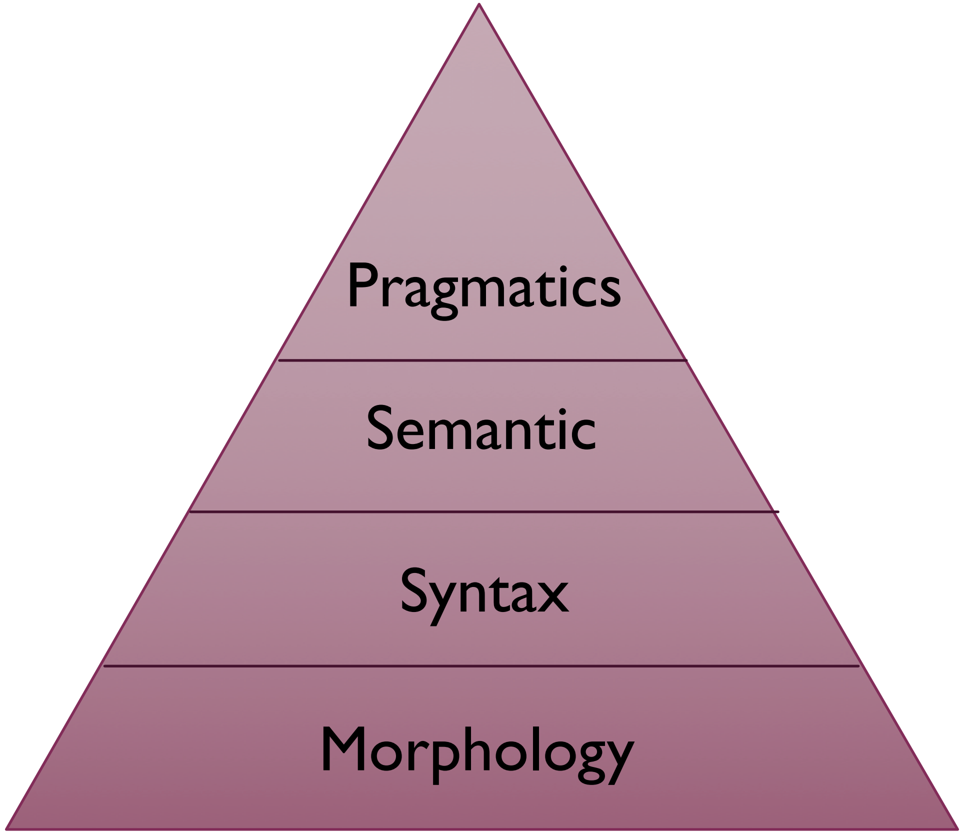
- Deep Learning Approach
- Long-term Sequences
- No feature generation
- Defining Model, Train and Inference



DEEP LEARNING VS. TRADITIONAL NLP

- WHY DL?
  - State-of-the-art performance in subtasks
    - Sentiment Analysis
    - MT
  - Look fancy and has upward trend
  - Most research is happening (ACL, EMNLP, arxiv, etc.)
- Our strategy?
  - Mainly work on DL approaches
  - However study traditional ones

NLP PYRAMID



MORPHOLOGY

- Analyses how words formed
- What is their origin
- Mostly deal with:
  - Prefixes/suffixes
  - Gender detection
  - Lemmatization
  - Spell checking
- Operations are at word level (viewed as a sequence of chars)

SYNTAX

- Underlying structure of a sentence
- Refer to grammar
- Most researched branch of CL
- Tasks:
  - POS tagging
  - Building Syntax Trees
  - Building Dependency Trees
- Usually works on sentences (viewed as sequence of words)

SEMANTIC

- Derives meaning from text
- Known problems
  - Name Entity Recognition
  - Relation Extraction
  - Semantic Role Labeling (shallow semantic parsing)
  - Word Sense Disambiguation
- Works on sentences (viewed as a sequence of words)

PRAGMATICS

- Analyses the text as a whole
- Tasks:
  - Summarization
  - Topic Segmentation
  - Coreference / Anaphora resolution (find out what word refers what.)
- Works on a text (viewed as a sequence of sentences)

Text Classification

SCENARIO OF TEXT CLASSIFICATION

- Task : Sentiment Analysis
- Input: IMDB movie reviews dataset
- Output: Polarity
- Pos example:
  - " I've seen this story before but my kids haven't.

    Boy with troubled past joins military, faces his past, falls in love and becomes a man. "
- Neg example:
  - " I feel about DARLING LILI. This massive musical is so peculiar and over blown, over produced and must have "

WHAT'S TEXT?

- A sequence of
  - Characters
  - ✓ **Words**
  - Phrases
  - Sentences
  - Paragraphs
  - …
- Seems natural to think of a text as a sequence of words!
- How to find the boundaries of words?
  - Punctuation or spaces

TOKENIZATION

- Task of chopping given a defined doc unit up into pieces, called **tokens**.
- A good example of simple whitespace tokenizer in python:
  - NLTK package for English :
    - nltk.tokenize.WhitespaceTokenizer,
    - nltk.tokenize.TreebankWordTokenizer()
    - nltk.tokenize.WordPuncTokenizer()
  - Hazm package for Persian:
    - word_tokenize()

TOKEN NORMALIZATION

- Stemming: Find the root form of the word called the **stem**
  - by removing and replacing suffixes
  - Porter's stemmer
- Lemmatization: Return the base of dictionary form of a word known **lemma**
  - WordNet lemmatizer
- Like:
  - wolves → wolf
  - talks → talk

PORTER'S STEMMER

- An algorithm for suffix stripping
- Using 5 heuristic phases for word reduction
- Applied sequentially
- Example of phase one rules:
  - SSES → SS caresses → caress
  - IES → I ponies → poni
  - SS → SS. caress → caress
  - S → cats → cats
- Problem with irregular forms, produce non-words

nltk.stem.PorterStemmer examples:

-feet → feet
-wolves → wolv
-cats → cats
-talked → talk

WORDNET LEMMATIZER

- Uses the WordNet database to lookup lemmas
- Only removes affixes if the resulting word is in its dic
- nltk.stem.WordNetLemmatizer
- Not all forms are reduced!

nltk.stem.WordNetLemmatizer examples:

-feet → foot
-wolves → wolf
-cats → cat
-talked → talked

OTHER NORMALIZATION

- Normalization capital letters
  - approaches:
    - Lowercasing the beginning of the sentence
    - Lowercasing words in titles
    - Leave mid-sentence words as they are
    - Use ML to retrieve **true casing**
- Acronyms
  - Frequently use in text msg and chats
  - eta, e.t.a., E.T.A. → E.T.A.
  - Write regular expressions, however it's hard!

BAG OF WORDS (BOW)

- Count the occurrence of a specific token in our text
- For each token we'll have a feature column called
  - **Text Vectorization**
- Example:
  - Sentence 1: I like to play tennis
  - Sentence 2: Did you go outside to play tennis
  - Sentence 3: John and I play tennis

Sentences	like	play	tennis	go	outside
#1	1	1	1	0	0
#2	0	1	1	1	1
#3	0	1	1	0	0

- Disadvantages?
  - Careful design in vocabulary
  - Sparsity
  - Discard word order in the context (This is interesting vs. Is this interesting?)

PRESERVE SOME ORDERING: N-GRAMS

- Solution: using **n-grams** technique
- Like: 2-grams for token pairs and so forth.
- Example:
  - Sentence 1: I like to play tennis
  - Sentence 2: Did you go outside to play tennis
  - Sentence 3: John and I play tennis

Sentences	like to	play tennis	tennis	go outside	outside
#1	1	1	1	0	0
#2	0	1	1	1	1
#3	0	1	1	0	0

- Disadvantages?
  - Too many text vectorizations

REMOVE SOME N-GRAMS

- There are 3 types of n-grams based on their occurrence in documents:
- High Frequency
  - Articles, prepositions, etc.
  - Called **Stop-words** → Remove them
- Low Frequency
  - Typos and rare n-grams → remove them
  - If we like to **overfit** our model, then it will be a good choice
- Medium Frequency
  - Good n-grams → Sustainable

TF-IDF

- **There are numerous medium frequency n-grams**
- Filtering out bad n-grams is useful
- Would be better if we set a ranking system for medium frequency n-grams
- Idea: n-grams with smaller frequency can be more discriminating
- Cause: capture a specific issue in the review

TERM FREQUENCY (TF)

- $tf(t, d)$
- Frequency for term or n-gram t in the document d
- Variants:

Weighting scheme	TF weight
binary	$0, 1$
Raw count	$f_{t,\,d}$
Term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
Log normalization	$1 + \log(f_{t,d})$

INVERSE DOCUMENT FREQUENCY (IDF)

- $N = |D|$ - total num of documents in corpus
- $|\{d \in D: t \in d\}|$ - num of documents where term t appears
- $idf(t, D) = \log \frac{N}{|\{d \in D: t \in d\}|}$

- TF-IDF:
  - $tfidf(t, d, D) = tf(t, d) . idf(t, D)$
- Achievement:

  We will find high frequent term in the given document

  but low frequent term in the whole collection of documents.

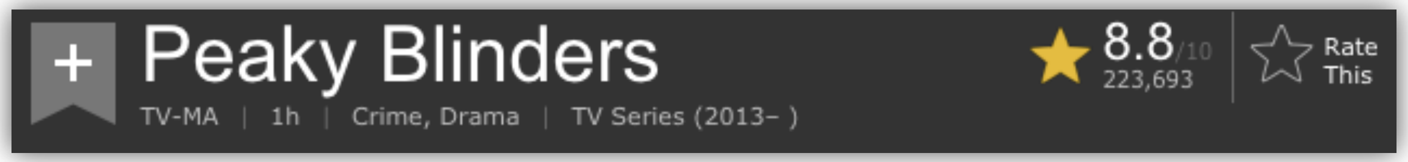
TF-IDF EXAMPLE

- Doc 1: I like to play tennis
- Doc 2: Did you go outside to play tennis
- Doc 3: John and I play tennis outside
- Replace counters with TF-IDF values
- tf-idf(like to, doc 1):
  - $1/3 \times \log(3/1) =$
  - $0.33 \times 0.47 \cong 0.15$
- tf-idf(outside , doc 2):
  - $1/3 \times \log(3/2) =$
  - $0.33 \times 0.176 \cong 0.058$

documents	like to	play tennis	tennis	outside
#1	0.15	--	--	--
#2	--	--	--	0.05
#3	--	--	--	--

LINEAR MODEL FOR CLASSIFICATION

- Task: Sentiment Classification
- Dataset: IMDB movie reviews dataset
- http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz



- 25000 reviews from IMDB
- Contain an **even** num of pos and negative reviews
- Randomly guessing yields, how many accuracy?
  - 50 percent
- A negative review has a score <= 4: label 0 and A positive review has a score >=7: label 1

BINARY LOGISTIC REGRESSION

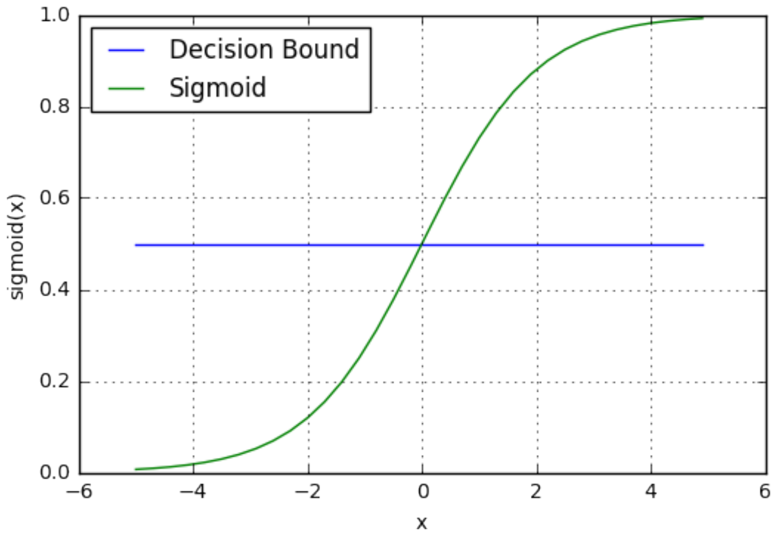
- Data split: 50/50
- Evaluation metric: accuracy
- Features: bag of 1-grams with TF-IDF values
- Extremely sparse feature matrix (25K rows and 74849 columns for TS)
- 99.8% are zeros
- Suggested Model for training: **Logistic Regression**
  - In statistics, uses to model probabilities of a certain class
  - Lose/win, pass/fail, alive/dead
  - **Linear** classification model
  - Handle sparse data
  - Weights can be interpreted

SIGMOID ACTIVATION FUNCTION

- In order to map predicted values to probabilities

- Maps any real values into another value between 0 and 1

  - $S(\zeta) = \dfrac{1}{1 + e^{-\zeta}}$

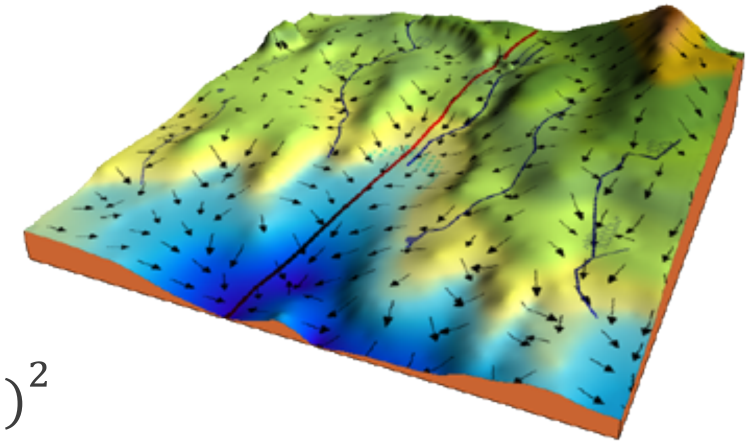


- $S(\zeta)$ = output between 0 and 1 (probability estimate)
- ζ = input to the function (your algorithm's prediction e.g. $mx + b$)
- e = base of natural log

- In order to map discrete class, we select a threshold like:
  - $P \geq 0.5, class = 1 \; and \; P < 0.5, class = 0$

GRADIENT DESCENT

- An optimization algorithm
- Find the parameters that minimize cost functions
- Parameters mean coefficients in linear regression and weights in neural networks
- In ML use it to update the parameters of our model.
- Iteratively moving in the direction of steepest descent
- Learning rate = step size
- E.g.: Given cost function:
  - $f(m,b) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$
- Gradient can be calculated from the partial derivatives.



GRADIENT DESCENT EXAMPLE

- We need to update **m** and **b** called **weights**
- $y = mx + b$
- $h(x) = mx + b$
- $h(x) =$ y, $h(x)$ called hypothesis in ML but this y is not actual value
- This is predicted y from our hypothesis
- Assumption: $b = 1 \; and \; m = 0.5 \; and \; x = 10$
- Then $h(x) = 0.5 \times 10 + 1$
- $predicted \; y \; is \; 6 \; but \; actual \; value \; is \; 5$
- $error = (h(x) - y)^2 = (6 - 5)^2 = 1$
- Expo 2 used to get rid of negative values
- Repeat for all data points in our DS and sum up all of them called **cost function**

X	Y
10	5
12	6.6
12	6.6
3	1
---	---

BETTER IMPROVEMENTS

- Add **2-grams** feature extraction (156821 columns)
- Throw away n-grams seen less than 5 times (min_frequency)
- Use special token like emoticons (;-))
- Adding stemming and lemmatization
- Apply different models
- Instead of using TF-IDF, use deep learning for word embedding

ANOTHER SCENARIO

- N-gram → feature index
- What if, when there are **2 TB** of texts?
- Problem on distributed systems
- Hold a very large vocabulary dictionary in memory
- So what's the solution?
  - Using hashing trick for squeezing
  - N-gram → hash(n-gram) % 2^b
  - Related paper: Feature Hashing for Large Scale Multitask Learning
  - https://arxiv.org/pdf/0902.2206.pdf

HASHING FEATURES

- Using a hashing function
- Maps any n-grams to a number range
- No need to save n-grams in a dictionary
- Might map any n-grams to the same number
  - Collision
- The larger range then less chance of collisions