# Synthesis

## Representations and language modeling

* Statistical Language Processing
  + Symbolic (rule-based): manipulation and processing of explicit symbols and rules
  + Stochastic: assign probabilities, data-driven (based on corpora), deep learning (ex.: HMM)
* Pattern matching: regular expressions: algebraic notation for characterizing a set of strings
  + Can be used to capture and substitute text
  + ELIZA: early NLP system with a series of cascade of regular expression substitution, matching and changing some part of the input lines
* Words:
  + Corpus: computer-readable collection of text
  + Word types: number of distinct words in a corpus, grouped in a set, the vocabulary (can be different word forms of a same lemma)
  + Word tokens: instances of types in the running text
* Tokenization: pre-process the text, segmenting into word tokens
  + Determines the vocabulary : the size of the vocabulary has a huge impact on the cost of computation. Word normalization can be used to reduce it
  + Simplest approach: Space-based
  + Deal with punctuation
* Word normalization: standard format for words
  + Upper/lower-cases: depend on the task: information retrieval dos not need upper-cases, but information extraction does
  + Lemmatisation: represent words as their lemma, morphological parsing (necessary for some languages)
  + Stemming: crudely cutting words
* Word distance:
  + Minimum edit distance: minimum number of editing operations between two strings - Insertion, Deletion, Substitution - needed to transform one into another (adapting costs: substitutions costs 2: Levenshtein distance)
    - Less costly way to find this distance is searching for the lest costly path of edits in a huge space
    - The edit distance between and
    - Algorithm for dynamic programming:
      * Initialize: and
      * Recurrence:
        + For from 1 to :

For from 1 to :

* Bag of words: unordered frequency count of words in vocabulary
  + Assumes that position does not matter, hence it is indifferent to syntax and semantics, and is only lexical a feature
  + Main goal: text classification: rules based on words, classifier with word frequencies as features
  + Also useful for document clustering, information retrieval
* Naive Bayes: naively assumes independence between words
  + Goal:
  + Bayes rule and the independence assumption:
* Practice: log-probabilities (add 1 to each count)
  + Training:
    - Create the vocabulary from documents
    - From :
      * For each class : compute the log-prior
      * For each word compute the log-likelihood
  + Inference
    - For each class
    - For each word : if
    - Return
* Document similarity
  + Cosine similarity:
* TF-IDF: remove frequent, but not significant words (stop words)
  + , count of word on document
  + , count of documents appears in, the total number of documnets
  + is the weight given to word in document
* Classification with Logistic regression:
  + Discriminative linear classifier: learns directly computing a linear score and applying a logistic function
  + Learn a vector and a bias to maximize the likelihood of making a good classification
  + Binary case: sigmoid
    - Extended to a multinomial case using a matrix W, a vector b and the softmax function
  + Maximize the likelihood by minimizing the cross-entropy:
  + Gradient descent
* For more complex NLP tasks
  + Segmentation: segment text into lexical units (words) (many possible uses for punctuation symbol, no typographic normalization, emoticons)
  + Lexical treatment: identify words (string to linguistic unit) and their properties, normalize text and deal with new words (with the help of morphology)
  + Syntax: constraints to obtain grammatically correct sentences (validity in position and agreement)
  + Semantics: meaning of statements, lexical (meaning of words) and compositional
  + Pragmatics: linked to the intent, statements are expected to be pertinent
* Solving ambiguities:
  + Ambiguity in written representation (speech synthesis)
  + Lexical ambiguity, on word grammatical properties (Part-of-speech tagging) or sense (Word sense disambiguation)
  + Syntactical ambiguity (parsing)
  + Ambiguity in interpreting a statement (sentiment classification, natural language inference)
* NLP applications: speech recognition, machine translation, grammar/spelling correction, ranking (assign larger probability to best option), optical character recognition, information retrieval, summarization, question answering, text generation
* Language model: estimates the probabilities of text sequences
  + Chain rule: decompose in smaller parts:
* Sequence of successive items: -gram
  + Ordered sequence looking words into the past
  + Depends on tokenization
  + Unigram (-gram), bigram (-gram), trigram (-gram)…
  + Markov assumption: probability depends upon a fixed number of words (the order): order k
    - Order 0 (unigram), order 1 (bigram), order 2 (trigram)…
    - Trigram is arguably the most used:
  + -gram are generally not sustainable due to the long-term dependencies, which are insufficient to the language model
  + -gram model is parametric, with the number of parameters : defined by the values of
* Model evaluation
  + Extrinsic evaluation: apply the model and use related metric
  + Intrinsic evaluation: (branching factor)
    - Perplexity is a simple function of the cross-entropy:
    - Maximizes likelihood, hence it minimizes perplexity
* Frequency is not the best measure of association between words
  + It is skewed: Zipf’s law:
  + Very frequent words are rarely the most useful for classification
  + Zero probability -grams: we use smoothing to solve it
    - Laplace Smoothing (add one): (avoids making never-seen sequences impossible)
    - Backoff: when not enough information at order back off to smaller orders
    - Interpolation: interpolate between smaller orders
* Pointwise Mutual information (PMI): evaluate how unexpected is the co-occurrence:
  + Negative values are unreliable

## Hidden Markov Models

* Stochastic process
  + Random state variable at time : or
  + Values of belong to the finite set
  + Probability for observing state at time :
  + Evolution process: from initial state , chain of state transitions ()
  + State sequence probability:
  + Model: transition probabilities + initial state probability
* Markov chain (discrete time)
  + Markov property (order ): limit dependencies
    - Bigram:
* Stationary Markov chain
  + Transitions do not depend on time:
  + Transition probability matrix
  + Initial probability vector
  + Constraints
* Model topology
  + Ergodic model (without constraints): is full
  + Lef-right model: is triangular
* Hidden Markov Model (HMM) for pattern recognition
  + Each class is represented by an HMM model
  + Combination of 2 stochastic processes (one observed and one hidden)
  + State sequence hidden
  + Observation generated by the states
* Discrete HMM
  + Set of Q discrete states
  + Set of observed symbols
  + Observation probability matrix (probability of observing each symbol in each state):
  + Defined by , and
* Continuous HMM
  + Probability of observing in state :
    - Gaussian mixture:
    - Gaussian model:
  + Defined by , , probability density function and
* HMM is a type of Dynamic Bayesian Network (DBN)
  + Independence of set of nodes:
  + Factorization:
* Generative HMM
* HMM decoding: assign the pattern to class
* Viterbi decoding to Part of Speech (POS) tagging
  + Let so search for then estimate likelihood by
  + Joint probability of best partial state sequence ending at on state and corresponding to the partial observation sequence :
    - Recurrence:
  + Algorithm:
    - 1st column: Initialization:
    - Columns 2 to :
      * Recursion
      * (Save best path)
    - Termination
    - Backtrack:
* Baum-Welch
  + - Backward variable:
    - Forward variable:
* Training: complete data
  + Given observation sequences and associated state sequences
* Training: incomplete data
  + Given observation sequences
  + Viterbi
  + Baum-Welch

## Structured prediction in natural language processing

* Structured prediction refers to the task of predicting structured objects rather than independent labels such as scalar values or categorical quantities
  + Not structured prediction: sentiment analysis, text classification,chat bots, autocorrection, speech recognition
  + Structured prediction: tagging, parsing, coreference resolution, text summarization
  + Depends: machine translation, natural language understanding, natural language inference
* Part of Speech (POS) tagging
  + English: of words are ambiguous
  + Example: Noun (N), Proper noun (PN), Transitive Verb (TV), Adjective (Adj), Determiner (D), Noun Phrase (NP: D + N), Verb Phrase (VP: TV + NP), Sentence (S: NP + VP)
  + Syntactic tree
  + Computational linguistics tasks: linguistic change, linguistic variation, comparison and control
  + NLP tasks: syntactic parsing, machine translation, sentiment analysis, text-to-speech
  + Algorithms: HMM, Conditional Random Fields (CRF), Maximum Entropy Markov Models (MEMM), Neural sequence models (RNNs or Transformers), Finetuned LLMs (ex: BERT)
    - Required a human-labeled training set
* Named Entity Recognition (NER)
  + Named Entity (people (PEO), location (LOC), organization (ORG), geo-political entity (GPE), dates, times, prices (AMOUNT))
  + Applications: sentiment analysis, question answering, information extraction
  + Algorithms for NER: Beginning-inside-outside (BIO) tagging, HMM, Conditional Random Fields (CRF), Maximum Entropy Markov Models (MEMM), Neural sequence models (RNNs or Transformers), Fine-tuned LLMs (ex: BERT)
    - Required a human-labeled training set
* Coreference resolution (CR): find all linguistic expressions in a given text that refer to the same entity
* Sequence labeler: model that assigns a label to each unit in a sequence, mapping a sequence of observations to a sequence of labels of the same length
* Goal: add information about the context in which a word appears
  + Conditional Random Fields (CRF) and Maximum Entropy Markov Models (MEMM) allow integration of rich features for better accuracy than HMM, however they require much slower training
* Dependency parsing: the syntactic structure of a sentence is described in terms of directed binary grammatical relations between the words
  + The arcs go from heads to dependents; the parse is a typed dependency structure
    - Dependency structure shows which words depend on (modify, attach to, or are arguments of) which other words
      * A dependency structure can be represented as a directed graph :
        + a set of vertices ( set of words in a given sentence + punctuation)
        + a set of ordered pairs of vertices (arcs)
      * Can have constraints: must be connected, must have root node, must be acyclic or planar
  + Computational linguistics tasks: Human communication, Linguistic variation
  + NLP tasks: Syntactic parsing for semantic parsing Machine translation, Sentiment analysis, Text-to-speech
  + Phrase attachment ambiguities
  + Three components: a stack, on which the parse is built; a buffer, containing the tokens to be parsed; a parser which takes actions on the parse via a predictor: an oracle, which requires supervised training for which it is necessary data
  + Algorithm:
    - The parser: walks through the sentence left-to-right, shifting items from the buffer onto the stack
    - At each time point: the top two elements on the stack are examined, the oracle makes a decision about what transition to apply to build the parse:
      * LEFTARC: assign the current word as the head of some previously seen word
      * RIGHTARC: assign some previously seen word as the head of the current word;
      * SHIFT: postpone dealing with the current word, storing it for later processing.
    - LEFTARC cannot be applied when ROOT is the second element of the stack.
    - LEFTARC and RIGHTARC require two elements to be on the stack to be applied.
* Dependency tree: directed graph that satisfies the following constraints:
  + Constraints
    - There is a single designated root node that has no incoming arcs
    - Appart from the root node, each vertex has exactly one incoming arc
    - There is a unique path from the root node to each vertex in V
  + ROOT: Root of the tree, head of the entire structure
  + NSUBJ: Nominal subject
  + OBJ: Direct object
  + NMOD: Nominal modifier
  + DET: Determiner
  + CASE: Prepositions, postpositions, other case markers
  + Universal Dependencies project
* Dependency treebanks: human annotated tree datasets
  + Bilexical affinities, Dependency distance, Intervening material,Valency of heads
* Evaluate parsing:
  + Exact match (EM): how many sentences are parsed correctly
  + Labeled attachment score (LAS): is a word properly assigned to its head with the correct dependency relation?
  + Unlabeled attachment score (UAS): is a word properly assigned to its head? (ignoring the dependency relation)
  + Label accuracy score (LS): what is the percentage of tokens with correct labels? (ignoring where the relations come from)

## Neural Language Models and Word Embeddings

* Difficulties with counting: models are huge, vectors are sparse
  + Use dense, distributed representations
    - Change the context for counting words: use surrounding words: co-occurrence matrix
    - Word meaning (distribution in the neighborhood): word embeddings: vector describing the distribution of other words in the neighborhood (cosine distance can be used to compute word similarity but it does not work well: all dimensions matter have the same weight)
  + Reduce the vocabulary (lemmatisation, TF-IDF)
  + Re-weight vectors
* Latent Semantic Analysis: Singular Value Decomposition (SVD):
  + diagonal matrix, with eigenvalues - ordered
  + , orthogonal; eigenvectors
  + Keep the first columns of , for the largest eigenvalues, to obtain embeddings
  + Very costly
  + New space is interpreted as topics
* Topic modeling: mostly generative models
  + Probabilistic LSA: generation of words follows a mixture of conditionally independent multinomial distributions, given topics:
    - relates to , relates to
  + Latent Dirichlet Allocation (LDA): topic distribution is assumed to have a Dirichlet prior
* -gram neural models
  + Teach a neural network to predict probability
  + Divided in two parts: processing the context words, obtaining output probability for the next word
* Neural Probabilistic Language Model (NLPM)
  + Continuous word vectors: Each input and output word is represented by a vector of dimension taking values in , rather than being discrete
  + Continuous probability function: The probability of the next word is expressed as a continuous function of the features of the word in the current context - using a neural network
  + Joint learning: The parameters of the word representations, and the probability function are learnt jointly
  + Projecting words: input one-hot encoded words to a densely connected to a smaller layer to smaller layer of dimension : the weight matrix are word embeddings
  + Obtaining scores:
    - Context is the concatenation of the smaller representation of words
    - Create hidden representation
    - Obtain scores for all words in given (output word embeddings to upscale the hidden representation to the predicted word dimension)
      * Compute probabilities: softmax
  + Training: minimize negative log-likelihood:
    - Equivalent to minimizing the Kullback-Leibler divergence
  + Updates:
    - First term increases the conditional log-likelihood of given
    - Second term decreases the conditional log-likelihood of all
    - Learning bottleneck due to softmax: Making hierarchical predictions: replace complexity in by ; Sampling based methods: sum over samples
* Learning word embeddings
  + Continuous bag of words (CBOW): simplifying architecture: ,
    - Parameters:
    - Training:
    - Objective function:
  + Skip-gram
    - Objective function:
    - Better handling of infrequent words
  + Avoid computing
    - Replace task by binary classification: predicting the right word
    - Only take into consideration the positive contribution:
    - Negative contribution by sampling wrong words:
      * Requires subsampling of frequent words in the noise distribution
  + GloVe: learn word embeddings by predicting word co-occurrence counts
    - Very fast training
    - Linear word representation relationships can capture meaning
  + Often reflect bias (gender bias)
  + Lexical ambiguity: polysemous, homonyms imply word embeddings with conflation
    - Solutions: sense embeddings, sparse coding, contextualized embeddings
* Prediction-based: fast and scale well with available data; dense and capture complex patterns; but require a lot of data and do not use all statistical information available
* Count-based: can also give dense representation (SVD to a PMI matrix); relatively fast; uses efficiently all information available and works well with little data
* Sub-word models: decompose to solve closure of vocabulary
  + Phonems, morphemes