**Sumary of Natural Language Processing**

**Regular Expression & Automata:**

1. **Regular Expression**

|  |  |
| --- | --- |
| *woodchuck* | /woodchuck/ |
| Avoid Case Sensitive | /[wW]oodchuck/ |
| /[A-Z]/ upper case letter | /[a-z]/ lower, /[0-9]/ single digit |
| Caret [^] negated | /[e^]/ either ‘e’ or ‘^’ /a^b/ pattern ‘a^b’ |
| /./ wildcard expression | Matches any single character /beg.n/ < begin, beg’n |
| *aardvark appears twice* | /aardva꙯꙯rk.\*aardvark/ |
| Anchors ^ start and $ end | /^The/ “The” at start, ꙯ $ for match a space at end |
| \b a word boundary | /\bthe\b/ matches ‘the’ not ‘other’ |
| \B matches a non-bounday | A word defined as digits, undersocres, letters in PC |

**Operator:**

|  |  |
| --- | --- |
| Disjunction /cat|dog/ | Match cat or dog |
| Precedence /gupp(y|ies)/ | Match guppy and guppies |
| Operator Precedence hierarchy | () > Counters\* + ? {} > anchors ^ $ > Disjunction | |
| Non-greedy match: \*? And \*+ | Matches as little as P. Default: greedy matching, |
| \d = [0-9] \D =[^0-9] | \w = [a-zA-Z0-9\_] = any alphanumeric \W = [^\w] |
| \s = [\_\r\t\n\f] whitespace |  |
| /{3}/ = exactly 3 occurrences | /{n,m}/ = n to m occurrences /{n,}/ at least n times |

**RE for Counting**

|  |  |
| --- | --- |
| Kleene \*: Zero or more | /[ab\*]/ “zero or more a’s or b’s” |
| Kleene +: One or more | /[0-9]+/ “a sequence of digits” |
| Optional elements /abs?/ | The preceding character ‘s’ or nothing |
| /{n}/ = exactly n occurrences | /{n,m}/ = n to m occurrences |
| /{n,}/ = at least n occurences | /{,m}/ = up to m occurrences |

**Special Characters**

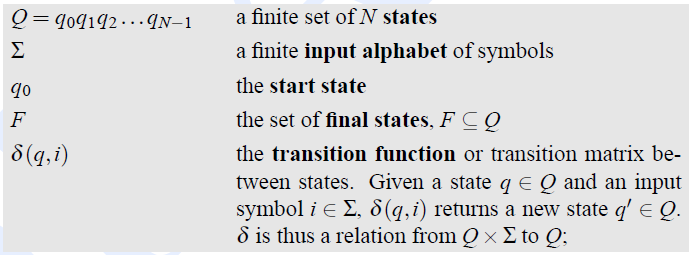
|  |  |
| --- | --- |
| **\\* = “\*” \. = “.” \? = “?”** | \n= “a new line” \t = “a tab” |

**Searching:** *the*

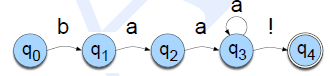
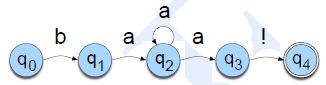
**False Positives**: incorrectly matched like “other or there” > **Precision**

**False Negatives:** incorrectly missed like “The” > **Recall**

2. Finite Automaton:



Left: Finite Right: Non-Finite-Automaton

**Three Solutions for Non-determinism:**

1. Backup, 2. Look-ahead, 3.Parallelism

**Chapter2:**

|  |  |
| --- | --- |
| **Text Normalization:** | **Tokenization, (stem)Lemmatization, sentence segmentation** |

*Stemming* chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. 

*Lemmatization* properly use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only.

|  |  |
| --- | --- |
| Type | An element of the vocabulary |
| Token | An instance of that type in running text |

**Morphology Analysis:**

**The study of the way words are built up from morphemes (stems and affixes.).**

|  |  |
| --- | --- |
| Inflectional:  Verbs | Combination of Stem and Affix: Cat and Cats  Stem+ s / +ing / +ed |
| Derivational: | Combination of Stem and a grammatical morpheme results in a new class: kill and killer, Stem /+ation /+ee /+er /+ness |
| Cliticization: | A unit whose status between an affix and a word  am > ‘m, are > ‘re … |

**FINITE STATE TRANSDUCER:**

**Chapter3 Language Models:**

**Markov Assumption: If a sentence “A B C D E”,** P( E | ABCD) = P(E|D)

|  |  |
| --- | --- |
| Unigram Model: P(A)=P(A) | P(ABCDE) = P(A)\* P(B)\* P(C)\* P(D)\* P(E) |
| Bigram Model: P(E|ABCD) = P(E|D) | P(CDE)=P(C)\*P(D|C) \*P(E|D) |
| Maximum Likelihood Estimate(MLE) | P(E|D)= count(D,E)/count(D) |

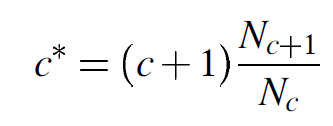
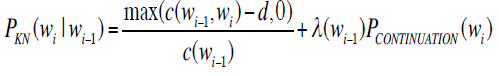
Practical Issues: log(P1\*P2\*P3) = logP1 + logP2 + logP3 (adding is faster than multiplying)

**Evaluation: Extrinsic and Intrinsic:**

Intrinsic Evaluation: 1.Perplexity(Inverse of probability of test), 2.Bad Approximation 3. Efficient

**Smoothing Method** for Solving Zero Counts(Sparseness):

|  |  |
| --- | --- |
| 1.Add-1 (Laplace) | Assume each word one more time than data. (#Zeros can’t be huge) |
| 2.Backoff and interpolation | Weighted probability estimates from all n-gram estimators. |
| 3.Good Turing Estimation | Use one time count to estimate word never seen. C\* see below: |
| 4.Kneser-Ney Smoothing | Absolute Discounting, d=0.75 |
| 1: text categorization  4. Most common | **Stupid backoff** helpful for very large web scale. |

SRILM

– hMp://www.speech.sri.com/projects/srilm/

KenLM

– hMps://kheafield.com/code/kenlm/

**Chapter4 Parts-of-Speech Tagging and Text Classification:**

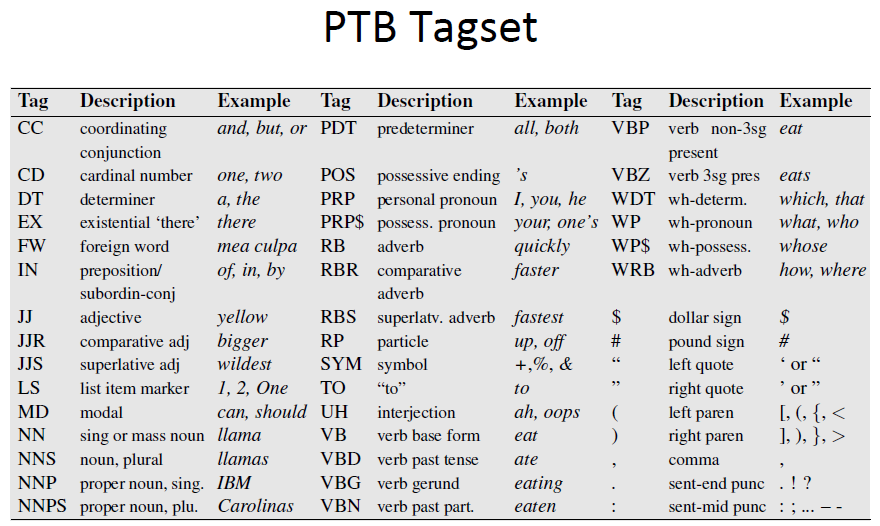
~9 Traditional **Word Classes**:

Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction

**Closed Classes:**

|  |  |
| --- | --- |
| Preposition | Occur before noun phrases: {on, under, over, near, by, at, from, to, with} |
| particle | Resemble prepositions or adverbs: {up, down, on, off, in, out, at, by} |
| Determiner | Before nouns: {a, an, the} |
| Conjunction | Join two phrases, clauses or sentences: {and, but, or, as, if, when} |
| Pronoun | A shorthand for some noun phrase: {she, who, I, others} |
| Auxiliary verb | Mark Some Semantic Features: {can, may, should} |
| numeral | one, two, three, first, second, third |

Tagset: 1.universal tagset: 17, 2. Brown Corpus: 87. 3. Penn Treebank (PTB): (45)



**Solving Ambiguity:**

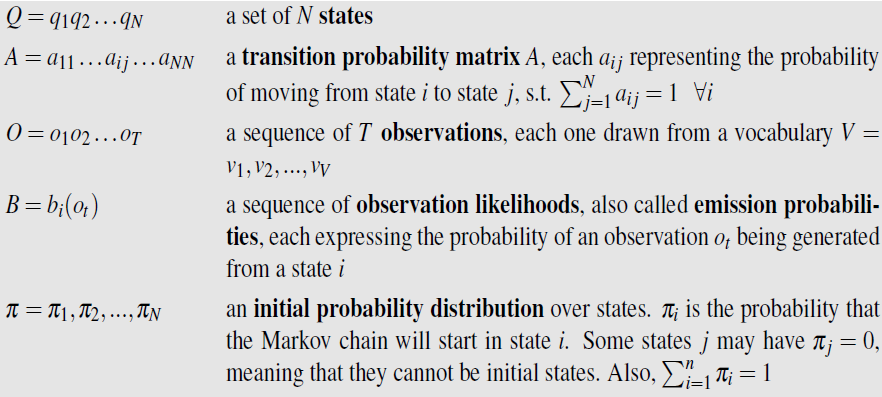
Rule-Based: based on lexical and other linguistic knowledge

Learning-Based: Trained on human annotated corpora like Penn Treebank

Statistical Models: Hidden Markov Model(**HMM**), Maximum Entropy, Conditional Random Field

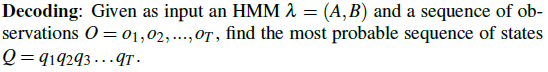
Rule Learning: Transformation Based Learning (TBL)

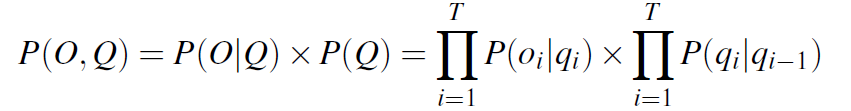
**Hidden Markov Model:**



For any model, such as HMM, that contains hidden variables:

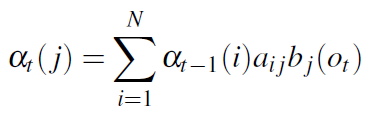
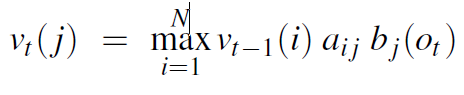
**Decoding:** Task of determining the hidden variables sequence corresponding to observations.



it takes N^k sequences.

Algorithm for Calculating likelihood for HMM:

|  |  |
| --- | --- |
| Forward Algorithm O(N^2T) | Summing up all the paths that lead to the current cell. |
| Viterbi Algorithm O(N^2T) | Calculates a max and keeps backpointers. |
| Beam Search | Viterbi, keep the best few hypotheses instead of whole column |

If **Unsupervised data, Forward-Backward Algorithm**

First, estimate a transition and emission probabilities. Then use it to derive better probabilities.

**Text Classification**

**Components of a probabilistic machine learning classifier:**

* Feature **Representation, 2.** A classification **function**, sigmoid and softmax
* **Performance Metric**, loss function to be mimnimized. “Cross-entropy Loss Function”
* **Algorithm** for optimizing the objective function. “Stochastic Gradient Descent”

**Discriminative Models:** Learn to classify, learn a probability over labels given data.

|  |  |
| --- | --- |
| Linear Regression | Z = **w**{\cdot}**x** +b |
| Logistic Regression | Unit in neural networks, y= alpha(z) = 1/(1+e^(-z)) |
| Loss Function | L(y\*,y) = 0.5(y\*-y)^2, | **Negative Log-likelihood (cross entropy loss)** |
|  | |
|  | |

**Stochastic Gradient Descent**

Stochastic gradient descent is called stochastic because it chooses a single random example at a time



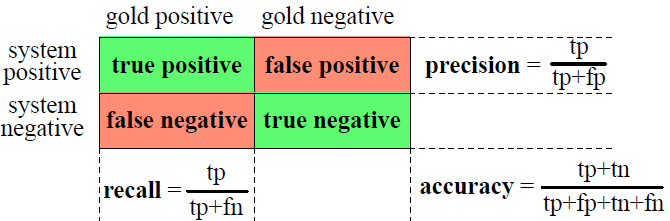
**Generative models:** Naive Bayes and HMMs, learn a probability distribution with given labels.

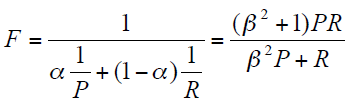
**Naïve Bayes Clasifiers**

|  |  |
| --- | --- |
|  | For a document d, out of all classes C, the classifier returns the class ^c which has the maximum posterior probability. |
|  | , and d = {f1,f2,…,fn} |
| Two Assumptions To Simplify | 1.Bag of words, 2. P(f1,f2,…,fn|c)=P(f1|c)\*P(f2|c)\*… P(fn|c) |

**Advantage: Fast, low storage, Robust to irrelevant features**

**Evaluation of Performance**



**Combined Measure:** **, if beta = 1,** 

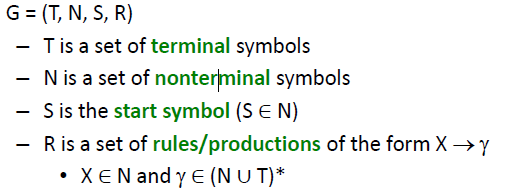
**For multi-classes:**

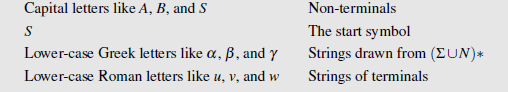
**Macroaveraging:** compute performance for each class, then average.

Microaveraging: Collect decision for all classes, compute contingency table.

**Formal Grammars & Syntactic Parsing**

**Context-Free Grammars**





**Formal Language:**

Cocke-Kasami Younger **CKY** Algorithm:

Chomsky Normal Form: the right-hand side is either two non-terminals or one terminal.

A > BC or A > a, CNF have binary sparse trees.

Three situations to **Convert CFG to CNF:**

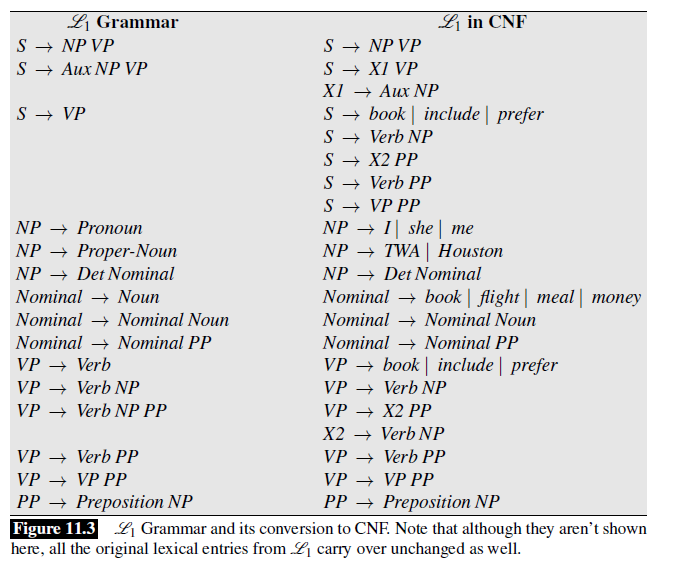
1. Rules that mix terminals and non-terminals in right-hand side.
2. Rules with a single non-terminal in the right-hand site
3. Rules with more than 2 non-terminals in the right-hand side.

Top-Down Parsing: The **Earley** Algorithm

Bottom-up, the **CKY** algorithm:

Partial Parsing:

Chunking: only need to identify phrase boundary and phrase type.

For a sentence of length n, work with u pper triangle of (n+1)\*(n+1)

Solving ambiguity, probability CFG, find most likely parse tree.

Problems: a) Poor independence assumption, b) Lack of lexical conditioning.

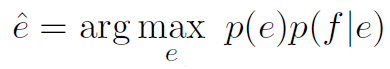
**Machine Translation (MT)**

**Classic MT:**

|  |  |  |
| --- | --- | --- |
| **Direct Translation** | 1.word-by-word, 2.no intermediate structures, except morpholo  3.Based on bilingual dictionary 4. reordering | Problems: No Parsing Component, Cannot Handle long-distance reordering. |
| **Transfer Models** | With constrastive knowledge: 1.Analysis and parse, 2.Transfer to parse tree, 3. Generation target sentence from parse tree | Requires a distinct set of transfer rules for each pair of languages. |
| **Interlingua** | 1. Parse into meaning representation , 2. Generate target from meaning | Requires exhaustive semantic analysis and disambiguation. |

**Statistical Machine Translation**

**Definition: Find the most likely target language sentence given a foreign source sentence.**



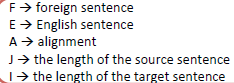
P(e) language model, estimates **fluency**, n-gram language model

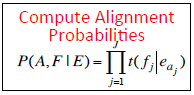
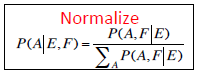
**p(f|e) translation model, faithfulness.**

**Spurious Word:** A word in foreign sentence doesn’t align with any word in English.

|  |  |  |
| --- | --- | --- |
| **Phrase-based Statistical MT** | 1.use phrases as fundamental units of translation. Translate rely on **translation matrix** and reorder reply on **distortion.** |  |
| **Neural MT** |  |  |

**Foreign Source(F), English Source(E), Alignment (A), and Length of the source sentence, I(length of target)**



Evaluation: **Bilingual Evaluation Understudy**

1.n-gram precision, 2. Ratio of correct n-grams to the total number of output n-grams. 3. Correct: number of n-grams shared with reference . Recall is ignored

Brevity Penalty: for translations that are shorter than the reference translations.

