* PART OF SPEECH TAGGING ⇒ MORPHOLOGICAL DISAMBIGUATION

- OPEN & CLOSED CLASSES:
  - OPEN CLASSES : NEW ONES CAN BE CREATED ALL THE TIME ex) NOUN, VERB, ADJ, ADV
  - CLOSED CLASSES HAVE A SMALL FIXED MEMBERSHIP ex) PREP, AUX, PRONOUN, FUNCTION WORDS.

- FOR POS TAGGING, A STANDARD SET OF TAGS MUST BE CHOSEN → PENN TREEBANK TAGSET (45 TAGS) COMMONLY USED.
  ⇒ POS TAGGING PROBLEM IS DETERMING THE POS TAG FOR A PARTICULAR INSTANCE OF A WORD.

- RULE-BASED TAGGING: ASSIGN ALL POSSIBLE TAGS TO WORDS FROM THE DICTIONARY
  → WRITE RULES BY HAND TO SELECTIVELY REMOVE TAGS
  → LEAVES THE CORRECT TAG FOR EACH WORD.
  - ENGCG / ENGTWOL TAGGING : RUN WORDS THROUGH FST MORPOLOGICAL ANALYZER TO GET ALL POS → APPLY NEGATIVE CONSTRAINTS.
- STOCHASTIC TAGGING : PROBABILISTIC SEQUENCE MODELS
  - HIDDEN MARKOV MODEL TAGGING (HMM) : SPECIAL CASE OF ~~BAYSE~~ BAYESIAN INFERENCE
    - USAGE: SPEECH RECOGNITION → OBSERVED: ACOUSTIC SIGNAL, HIDDEN: WORDS
      HANDWRITING RECOGNITION → OBSERVED: IMAGE, HIDDEN: WORDS
      POS TAGGING → OBSERVED : WORDS, HIDDEN: POS
      MT → OBSERVED: FOREIGN WORDS, HIDDEN: WORDS IN TARGET LANG

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n / w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \underbrace{P(w_i | t_i)}_{\text{likelihood}} \underbrace{P(t_i | t_{i-1})}_{\text{prior}}$$

- TAG TRANSITION PROBABILITIES $p(t_i | t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$

- WORD LIKELIHOOD PROBABILITIES $p(w_i | t_i) = \frac{C(t_i, w_i)}{C(t_i)}$

⇒ A MARKOV CHAIN IS A SPECIAL CASE OF A WFST, IN WHICH THE INPUT SEQUENCE UNIQUELY DETERMINES WHICH STATES THE AUTOMATON WILL GO THROUGH.
→ HIDDEN MARKOV MODEL IS AN EXTENSION OF A MARKOV CHAIN IN WHICH THE INPUT SYMBOLS ARE NOT THE SAME AS THE STATES.

- VITERBI ALGORITHM: CREATE AN ARRAY WITH COLUMNS CORRESPONDING TO INPUTS AND ROWS CORRESPONDING TO POSSIBLE STATES → SWEEP THROUGH USING TRANSITION PROBS & OBSERVATION PROBS → STORE ONLY MAX PROB PATH TO EACH CELL.
- EVALUATION OF POS TAGGING:
  - COMPARING WITH MANUALLY CODED "GOLD STANDARD"

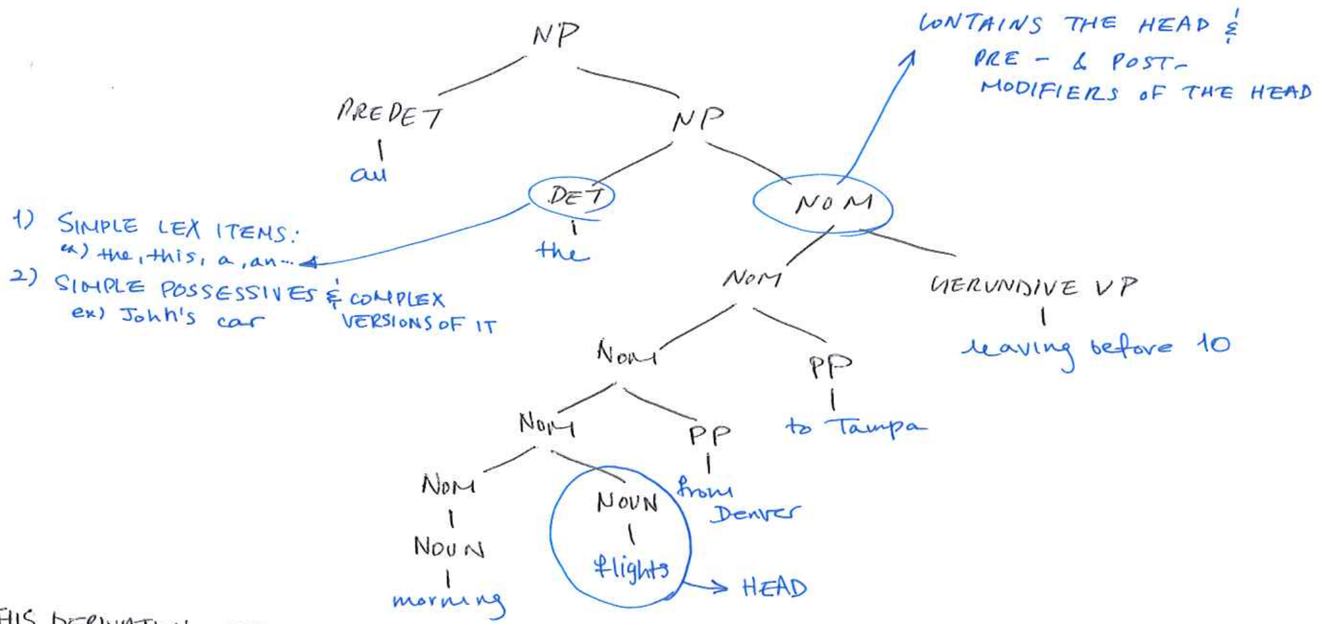
* SYNTAX → THE WAY WORDS ARE ARRANGED TOGETHER.

- CONSTITUENCY : GROUPS OF WORDS MAY BEHAVE AS A ~~SINGLE~~ SINGLE UNIT OR PHRASE
  ⇒ A CONSTITUENT
  - CONSTITUENTS CAN BE SHOWN TO BEHAVE IN SIMILAR WAYS WITH RESPECT TO THEIR INTERNAL STRUCTURE & OTHER UNITS IN THE LANGUAGE.
- CONTEXT-FREE GRAMMARS (CFG) : MATHEMATICAL SYSTEM FOR MODELLING CONSTITUENT STRUCTURE.
  - TERMINAL SYMBOLS: CORRESPOND TO WORDS IN THE ~~LANGUA~~ LANGUAGE
  - NON-TERMINAL SYMBOLS: EXPRESS CLUSTERS OR GENERALIZATIONS
  - RULES: EQUATIONS THAT CONSIST OF A SINGLE NON-TERMINAL ON THE LEFT & ANY NUMBER OF TERMINALS & NON-TERMINALS ON THE RIGHT

    ex) NP → Det Nominal
    NP → Proper Noun
    Nominal → Noun | Nominal Noun

    } RULES FOR NOUN PHRASES.

    → EXPLICIT DISJUNCTION - TWO KINDS OF NOMINALS
    → RECURSIVE DEFINITION - SAME NON-TERMINAL ON THE RIGHT & LEFT-SIDE OF THE RULE.
  - DERIVATION: A SEQUENCE OF RULES APPLIED TO A STRING THAT ACCOUNTS FOR THAT STRING → COVERS ALL, BUT ONLY, THE ELEMENTS IN THE STRING
- SENTENCE TYPES: DECLARATIVES, IMPERATIVES, YES-NO QUESTIONS, WH-QUESTIONS.
- NOUN PHRASES:



→ THIS DERIVATION OVERGENERATES, AS IT DOES NOT CARE ~~ABOUT~~ ABOUT AGREEMENTS
  ex) DOES NOT DISTINGUISH B/W THIS FLIGHT & THESE FLIGHT

- VERB PHRASES: CONSIST OF A HEAD VERB ALONG WITH 0 OR MORE FOLLOWING CONSTITUENTS (ARGUMENTS)
  - SUBCATEGORIZE THE VERBS IN A LANGUAGE ACCORDING TO THE SETS OF VP RULES THAT THEY PARTICIPATE IN.

- CFGs PROVIDE INFORMATION ABOUT THE BASIC SYNTACTIC STRUCTURE BUT OVERGENERATE.
  ⇒ THERE ARE WAYS TO DEAL WITH THE PROBLEM, BUT NOT ELEGANT
  ⇒ THERE ARE SIMPLER & ELEGANT SOLUTIONS OUTSIDE OF THE CFG FRAMEWORK

* TREEBANKS : CORPORA IN WHICH EACH SENTENCE HAS BEEN PAIRED WITH A PARSE TREE
  → IMPLICITLY DEFINE A GRAMMAR FOR THE LANGUAGE COVERED IN THE TREEBANK.
    · THE DERIVED GRAMMAR IS VERY FLAT AS THEY TEND TO AVOID RECURSION
  - HEAD FINDING: USE A SIMPLE SET OF TREE TRAVERSAL RULES SPECIFIC TO EACH NON-TERMINAL IN THE GRAMMAR
  - PARSING: GIVEN A STRING OF TERMINALS & A CFG, DETERMINE IF THE STRING CAN BE GENERATED BY THE CFG → RETURN THE PARSE TREE(S) FOR THE STRING
    · TOP-DOWN PARSING: START SEARCHING SPACE OF DERIVATIONS FOR THE START SYMBOL
    · BOTTOM-UP PARSING: START SEARCH SPACE OF REVERSE DERIVATIONS FROM THE TERMINAL SYMBOLS IN THE STRING
      → NEVER EXPLORES OPTIONS THAT WILL NOT LEAD TO A FULL PARSE, BUT CAN EXPLORE MANY OPTIONS THAT NEVER CONNECT TO THE ACTUAL SENTENCE
      → NEVER EXPLORES OPTIONS THAT DO NOT CONNECT TO THE ACTUAL SENTENCE, BUT CAN EXPLORE OPTIONS THAT CAN NEVER LEAD TO A FULL PARSE.
    ⇒ DYNAMIC ALGORITMS BASED ON BOTH APPROACHES ACHIEVE $O(n^3)$ RECOGNITION TIME, WHERE n IS THE LENGTH OF THE INPUT STRING
      → CACHING IS CRITICAL TO OBTAINING A POLYNOMIAL TIME PARSING ALGORITHM FOR CFGs
  - CKY ALGORITHM: PRODUCES ALL POSSIBLE PARSE TREES
    · GRAMMAR MUST BE CONVERTED TO CHOMSKY NORMAL FORM (CNF) → LEXICAL RULES: MUST HAVE EITHER EXACTLY 2 NON-TERMINAL SYMBOLS ON RHS OR 1 TERMINAL SYMBOL
    · PARSE BOTTOM-UP STORING PHRASES FORMED FROM ALL SUBSTRINGS IN A TRIANGULAR TABLE (CHART)
    · COMPLEXITY:

$$(n(n+1)/2) = O(n^2) \text{ cells} \times O(n) \text{ possible split points}$$
$$= O(n^3)$$

    → SYNTACTIC AMBIGUITY CAN BE RESOLVED BY USING PROBABILISTIC CFG (PCFG) TO RUN CKY ALGORITHM TO DETERMINE THE MOST LIKELY PARSE TREE.

* DEPENDENCY PARSING
  - DEPENDENCY SYNTAX: ASSUMES THAT SYNTACTIC STRUCTURE CONSISTS OF LEXICAL ITEMS LINKED BY BINARY ASYMMETRIC RELATIONS CALLED DEPENDENCIES
    → HEAD RULES CAN BE USED TO EXTRACT A DEPENDENCY PARSE FROM A CFG PARSE.
  - SOURCES OF INFORMATION FOR DEPENDENCY PARSING:
    · BILEXICAL AFFINITIES
    · DEPENDENCY DISTANCE
    · INTERVENING MATERIAL
    · VALENCY OF HEADS: HOW MANY DEPENDENTS ON WHICH SIDE ARE USUAL FOR HEADS?
  - MaltParser: a simple form of greedy discriminative dependency parser
    → does a sequence of bottom-up actions
    · Maintains 4 stacks:
      - σ: starts with ROOT → word left between Root & . is the Root
      - β: starts with the input sentence → terminates when this is empty.
      - A : a set of dependency arcs A → starts empty
      - Set of actions
    ⇒ Provides very fast linear time parsing

- PROJECTIVITY: dependencies from a CFG tree using heads, must be projective.
  → there must not be any crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words.

* PHRASE CHUNKING: find all non-recursive NPs and VPs in a sentence.
  → tag individual words with one of 3 tags:
  - B (begin): word that starts a new target phrase
  - I (inside): word that is part of target phrase but not the first word.
  - O (other): word that is not part of target phrase.

- Evaluating chunking:

$$\text{Precision} = \frac{\text{\# of correct chunks found}}{\text{Total \# of chunks found}}$$

$$\text{Recall} = \frac{\text{\# of correct chuncks found}}{\text{Total \# of actual chunks}}$$

→ F measure: harmonic mean of the two.

* Machine Translation

- Rule-Based: Hand-written transfer rules
  Rules can be based on lexical / structural transfer
  ⊕ Firm grip on complex translation phenomena
  ⊖ often very labor-intensive → lack of robustness
- Statistical: Mainly word or phrase-based translations
  translations are learned from actual data
  ⊕ Translations are learned automatically
  ⊖ Difficult to model complex translation phenomena.
  → Challenges:
    - Sparsity: training models need a lot of data
      genre & domain sensitive
      worse for language with rich morphology
    - Hallucinations: cause unintensionally ~~bais~~ biased translations
- Automatic Evaluation: BLEU METRIC
  - Modified n-gram precision with length penalty
  - quick, inexpensive & language independent
  - correlates highly with human evaluation
  - bias against synonyms & inflectional variations.
- Word ~~Al~~ Alignment:
  - given parallel corpus, with word alignment → compute translation rules
  - Given translation rules → compute probability of translation.
  - Mapping alignment: given source word position i and target word position j,
    $a: i \to j$ ⇒ $a: \{1 \to 1, 2 \to 3, 3 \to 2, 4 \to 4\}$
- Word-based translation model:
  Initialize translation rules
  Do until (convergence) {
    Translate parallel corpus
    Count translations
    Create translation rules
  }
  ⇒ Does not take into account:
  - Dropping words
  - Inserting
  - word order
  - one-to-many translation.
- Phrase-based translation model: extract phrase-pair consistent with word alignment
  ⇒ all words of the phrase-pair have to align to each other and no other word.
  - Collect all phrase-pairs from the data
    → Calculate probabilities: $t(e|f) = \frac{\text{count}(e, f)}{\sum_i \text{count}(e_i, f)}$
    translation probability
    target sentence
    source sentence
- Evaluation:
  - Automatic Metrics:
    - goals: low cost, tunable, meaningful, consistent, correct
      automatically optimize system performance towards metric.
    - basic strategy: given machine translation output & human reference translation, compute the similarity between them.
    - Using precision & recall to compute harmonic ~~F~~ average ⇒ F-measure.
      → FLAW: no penalty for re-ordering

- Word Error Rate : minimum number of editing steps to transform output to reference
  - Match : no cost
  - substitutions , insertion , deletion :

$$\text{Levenshtein distance (WER)} = \frac{\text{substitutions + insertions + deletions}}{\text{reference length}}$$

- BLEU : compute precision for n-gram of size 1 to 4
  - penalizes short translations → brevity penalty
  - over entire test set, not individual sentences.

$$\text{BLEU} = \min\left(1, \frac{\text{output length}}{\text{reference length}}\right)\left(\prod_{i=1}^{4} \text{precision}_i\right)^{1/4}$$

  - multiple references can be used
    → n-grams may match any of the references.
- Critique of Automatic Metrics : Ignores relevance of words,
  operates on local level → donnot consider overall grammaticality/ sentence meaning
  Scores are meaningless → test-specific
  Human translators score low on BLEU

– PHRASE – BASED MODELS
- Advantages : many-to-many translation can handle non-compositional phrases
  use of local context in translation
  → the more data, the longer phrases can be learned.
- Phrase translation table : table with phrase translations & their probabilities.
  ↳ will include lexical variations, morphological variations, include function words and noise.
- Tuning : the process of finding the optimal weights for the linear decoding model
  → optimal weights are those which maximize translation performance on a small set of parallel sentences (tuning set)
  - MERT (Minimum Error Rate Training) : decodes the whole training set and generates an n-best list → model weights updated based on this decoder output
    → optimization process repeats until some convergence criterion is met

  transforming the source sentence into target sentence
  ⇒ find e_{best} with highest probability.
- Methods of search space reduction :
  - Recombination : when two hypothesis paths lead to two matching hypotheses ; if same number of foreign words are translated and same target words are in the output with different scores, drop worse hypothesis
  - Pruning : remove bad hypotheses early by putting ~~competi~~ comparable hypothesis into stacks & limiting # of hypotheses in each stack
    ↳ that have translated same # of input words.
  ⇒ Quadratic complexity

- Tokenization : solution for SMT being not as effective for language with rich morphology
  → break up the words to symmetrize source & target language.

* LEXICAL SEMANTICS

- Word sense : a discrete representation of an aspect of a word's meaning
- Homonyms : words that share a form but have unrelated, distinct meanings.
  ↳ homographs (written same, read different), homophones (written different, read same)
- Polysemy : a polysemous word has <u>related</u> meanings
  → lots of types of polysemy are systemetic.   ex) school, hospital ...
  ⇒ building ⟷ organization
  ⇒ Do "zeugma" test to find out if a word has more than one sense.
  ex) Does Lufthansa serve breakfast & San Jose?
- Synonyms : two lexemes are synonyms if they can be substituted for each other in all situations ⟹ they have the same propositional meaning
  ⟷ Antonym
- Hyponymy : one sense is hyponym of another if the first sense is more specific, denoting subclass of the other.   ⇒ IS - A hierarchy
  ⟷ Hypernymy

ex) $\text{vehicle} \underset{\text{hyponym of}}{\overset{\text{hypernym of}}{\rightleftarrows}} \text{car}$ ⇒ hyponymy is <u>transitive</u>

* Similarity algorithms

- thesaurus-based algorithms – are words "nearby" in hypernym hierarchy? – do they have similar glosses?
- Distributional algorithms – do words have similar distributional contexts?

$$Sim_{path} = \frac{1}{pathlen(c_1, c_2)}$$

- Path based similarity : two concepts are similar if they are near each other in the thesaurus hierarchy (concepts have path 1 to themselves)
  → problem: assume each link represents a uniform distance.
- Information Content Similarity :
  - $P(c)$: the probability that a randomly selected word in a corpus is an instance of concept c → $P(root) = 1$

$$P(c) = \frac{\sum_{w \in words(c)} count(w)}{N}$$

words(c) : set of all words that are children of node C

N → total number of words

  - Information content (IC) $= -\log P(c)$
  - Lowest Common Subsumer (LCS – most informative subsumer) :
    LCS (c_1, c_2) = the most informative (lowest) node in the hierarchy subsuming both c_1 and c_2

$$Sim_{resnik}(c_1, c_2) = -\log P(LCS(c_1, c_2))$$

→ the IC of the most informative subsumer of the two nodes.

$$Sim_{Lin}(c_1, c_2) = \frac{2\log P(LCS(c_1, c_2))}{\log P(c_1) + \log P(c_2)} \propto \frac{IC(common(c_1, c_2))}{IC(description(c_1, c_2))}$$

→ the more differences b/w c_1 and c_2, the less similar they are

amount of information needed to state the commonality

information needed to fully describe what A & B are.

$$\text{Sim}_{eLesk}(c_1, c_2) = \sum_{r,q \in RELS} \text{overlap}(\text{gloss}(r(c_1)), \text{gloss}(q(c_2)))$$

→ two concepts are similar if their glosses contain similar words

→ compute overlap for other relations (RELS) as well ⇒ glosses of hypernyms & hyponyms

- Distributional models of meaning (vector-space models of meaning)
  → offer much higher recall than hand-built thesauri (but lower precision)
  - Term-document matrix: each cell is count of term t in a document d: $tf_{t,d}$
    → each document is a count vector in $\mathbb{N}^V$ → two documents are similar if their vectors are similar
    → each word is a count vector in $\mathbb{N}^D$ → two words are similar if their vectors are similar.
  - Term-context matrix: use smaller contexts (paragraph, sentences) instead of entire documents

  ⇒ Instead of using raw counts: term-document matrix → TF-IDF (Term Frequency - Inverse Document Frequency)

  Term-context matrix → Positive Pointwise Mutual Information (PPMI)

- Pointwise Mutual Information: do events X and Y co-occur more than if they were independent?

$$PMI(X, Y) = \log_2 \frac{P(X,Y)}{P(X)P(Y)}$$

⇒ For PPMI, replace all negative values with zero.

* Information Retrieval:

- Indexing: mapping between terms & documents.
  → "query" is a Boolean expression over vectors.
    ↳ For boolean retrieval, term-document matrix is binary ⇒ marks whether the term appears in the document or not.

  ⊕ Accurate with right strategies

  ⊕ Efficient for the computer

  ⊖ Results too many or none.

  ⊖ Users must know boolean logic

  ⊖ doesn't account for the fact that words have multiple meanings

  - Indexing counts: entries are counts of occurrences of a term in a document.

- Document frequency: documents are most likely described well by rare terms that occur in them frequency
  → High Term Frequency (TF) is evidence of meaning
  → Low Document Frequency (DF) is evidence of importance.
    ⇒ High Inverse Document Frequency (IDF)

  ⇒ Term Weight = TF-IDF (product of TF & IDF)

$$W_{t,d} = tf_{t,d} \times \log \frac{N}{df_t}$$

← # of documents in which "t" appears (normalized, inverted & scaled)

* TEXT SUMMARIZATION :

- simple baseline: take the first sentence
- 3 stages: 1) content selection: choose sentences to extract from the document
  2) information ordering: choose an order to place them in the summary ↳ baseline: use order of appearance in doc
  3) sentence realization: clean up the sentences. → baseline: use original of sentences
- CONTENT SELECTION:
  - UNSUPERVISED CONTENT SELECTION: CHOOSE SENTENCES THAT HAVE SALIENT / INFORMATIVE WORDS
    - → tf-idf : $w_i = tf_{ij} \times idf_i$
    - → TOPIC SIGNATURE : CHOOSE A SMALLER SET OF SALIENT WORDS
      - MUTUAL INFORMATION
      - LOG-LIKELIHOOD RATIO (LLR):

$$w_i = \begin{cases} 1 & \text{if } -2\log\lambda(w_i) > 10 \\ 0 & \text{otherwise} \end{cases}$$

  - SUPERVISED CONTENT SELECTION:
    GIVEN A LABELED TRAINING SET OF GOOD SUMMARIES FOR EACH DOCUMENT, (hard to get)
    ALIGN THE SENTENCES IN DOC WITH SENTENCES IN THE SUMMARY (difficult)
    ⇒ EXTRACT: POSITION OF SENTENCE, LENGTH OF SENTENCE, WORD INFORMATIVENESS, COHESION
    ⇒ TRAIN A BINARY CLASSIFIER (PUT SENTENCE IN SUMMARY?)
    → performance not better than unsupervised.
- ROUGE (Recall Oriented Understudy for Gisting Evaluation): based on BLEU
  INTRINSIC METRIC FOR AUTOMATICALLY EVALUATING SUMMARIES.

  GIVEN DOCUMENT D, AUTOMATIC SUMMARY X,

$$\text{ROUGE}_{2} = \frac{\sum_{S \in \{RefSum\}} \sum_{\text{bigrams } i \in S} \min(\text{count}(i, X), \text{count}(i, S))}{\sum_{S \in \{RefSum\}} \sum_{\text{bigrams } i \in S} \text{count}(i, S)}$$

bigrams

human produced