

Summary – Natural Language Processing

V 2.6

Index

Introduction.....	7
NLP applications.....	7
Major topics.....	7
Techniques.....	7
How it works.....	8
Ambiguity.....	8
Solution.....	8
Words.....	9
Morphology.....	9
Inflection and derivation.....	9
Nouns and verbs.....	9
Words and Transducers.....	9
Error Correction, Prediction, N-Grams.....	11
Error Correction.....	11
Bayesian model and noise channel.....	11
Minimum edit distance.....	12
N-grams.....	12
Applications.....	12
Counting.....	12
Language modeling.....	12
Shannon's Method.....	13
Evaluation.....	13
Smoothing.....	13
Zero counts.....	13
Semantic.....	14
Representation language.....	14
Language structure.....	15
Verb.....	15
Representation language types.....	15
Semantic Networks.....	15
Logic.....	16
FOL.....	16
Description logic.....	17
Frame-based representation.....	17
Tools.....	18
Production-rule Systems.....	18
Prolog.....	18
Ontologies.....	18
OWL web ontology language.....	19
SPARQL.....	20
Top and domain ontology.....	20

Tools.....	21
Semantic Analysis.....	21
Syntax-based semantic analysis.....	21
Syntactic representations.....	21
Semantic augmentation.....	22
Idioms.....	22
Semantic Grammars.....	22
Information Extraction.....	23
NER Named Entity Recognition.....	23
Lexical semantics.....	24
Lexeme and Lexicon.....	24
Lexemes relationships.....	24
Polisemy/Homonymy.....	24
Polisemy.....	24
Homonymy.....	25
Synonymy.....	25
Antonymy.....	25
Hyponymy/Hypernymy.....	25
Hypernymy.....	25
Hypernymy.....	25
Meronymy/Holonymy.....	25
Meronymy.....	25
Holonymy.....	25
Lexical Database.....	26
Wordnet.....	26
Structure.....	26
WordNet translation.....	27
Lexical DB and NLP.....	27
Internal structure of words.....	27
Word Sense Disambiguation (WSD).....	28
Selectional Restrictions.....	28
Machine Learning.....	28
Supervised Machine Learning.....	29
POS tagging, syntactic analysis.....	30
Part of Speech tagging.....	30
Part of Speech.....	30
Open and Close classes.....	30
POS tagging.....	30
HMM POS tagging.....	31
Markov chain.....	31
Viterbi algorithm.....	31
Formal Grammars.....	32
Grammar (or Syntax).....	32
Constituency.....	32
Context Free Grammars (CFG).....	32
Parsing.....	32
English Grammar Example.....	32
Sentences.....	32
Noun phrases.....	33
Verb phrases.....	33

Problems:.....	33
Probabilistic approaches.....	34
Treebanks.....	34
Dependency grammars.....	34
Grammars and Parsing.....	34
Shallow parsing.....	34
Probabilistic parsing.....	34
Dependency Grammars.....	35
Parsing.....	35
Syntax: tools (parsing, corpora).....	36
Parsing.....	36
Classical CFG.....	36
Stochastic.....	36
PCFG.....	36
L-PCFG.....	36
Others.....	37
Dependency grammar.....	37
Corpora.....	37
How corpora are built.....	37
Bootstrap.....	37
Kappa measure.....	38
Syntax: useful stochastic models.....	38
Generative.....	38
HMM (look before).....	38
Discriminative.....	38
MaxEnt.....	38
Entropy.....	38
MaxEnt Markov Models (MEMM).....	40
CRF.....	40
Generative vs discriminative.....	41
Training and Cross-validation.....	41
Model evaluation.....	41
POS Tagging and Chunking.....	41
POS tagging.....	41
Stochastic approaches.....	42
HMM.....	42
Chunking (shallow parsing).....	43
Voice.....	44
Speech Analysis.....	44
Human Voice: prosody and voice descriptors in DA.....	44
Time and frequency domain.....	44
Vowels and consonants.....	45
Vowels.....	46
F1 F2 space.....	46
Bark scale.....	46
Vowels normalization.....	47
Consonants.....	47
Perception: prosody and voice.....	48
Perception process.....	48
Focus and attention.....	49

Language perception.....	49
Cognitive theories.....	49
Prosody: prosody and voice descriptors in digital audio.....	49
Intonation.....	50
Autosegmental Metrical Theory (AM).....	50
Tones and Break Indices (ToBI).....	50
Coherence.....	52
Paratone.....	52
Tempo and volume.....	52
Stress and Rhythm.....	52
Stress.....	52
Rhythm.....	53
Paralanguage.....	53
Phrasing.....	53
Speech Perception.....	53
Speech processing.....	54
Intro.....	54
Phonetics and phonology.....	54
Human voice in the frequency domain.....	54
Text-To-Speech (TTS).....	55
Automatic Speech Recognition (ASR).....	56
Steps.....	57
1. Short time Fourier transform.....	57
2. Mel filter bank.....	58
3. Cepstrum.....	58
4. Hidden Markov Model for ASR.....	58
Recap.....	61
Pragmatics.....	61
Dialogue and conversational agents.....	61
Linguistics of Conversation.....	61
Basic Conversational Agents.....	62
Turn-taking.....	62
Speech Acts.....	62
Grounding.....	63
Conversational Structure.....	63
Implicature.....	63
Dialogue System Architecture.....	64
Speech recognition (ASR).....	64
Natural Language Understanding (NLU).....	64
Natural Language Generation.....	64
Text to speech synthesis.....	64
Dialogue Manager.....	64
VoiceXML.....	65
Information state.....	65
Dialogue-act interpreter.....	66
Correction.....	66
Dialogue-act generator.....	67
Confirmation.....	67
Rejection.....	67
Evaluation.....	67

Paradise evaluation.....	67
Plan based dialogue agents.....	68
Utility based.....	68
Policy strategy.....	68
Markov Decision Process (MDP).....	68
Planning.....	70
Plan inferential interpretation and production.....	70
Computing with Affective Lexicons.....	71
Sherer affective states.....	71
Attitudes/Sentiment.....	71
Emotion.....	72
Lexicons for detecting document affect.....	72
Personality traits.....	73
Interpersonal stance.....	73
Chatbots and Conversational agents.....	73
Different types of chatbots.....	73
Discourse.....	74
Anaphora Resolution.....	74
Reference.....	74
Reference Phenomena.....	74
Coherence.....	75
Discourse Structure.....	75
Rhetorical structure.....	75
Summarization and QA.....	76
Summarization.....	76
Single document.....	77
Content Selection.....	77
Unsupervised content selection.....	77
Supervised content selection.....	78
Multiple Documents.....	78
Content selection.....	79
Information ordering.....	79
Content selection + Information ordering (HMM).....	79
Focused Summarization.....	79
Information Extraction.....	79
Summarization Evaluation.....	80
Neural Networks and NLP.....	80
Neural Networks.....	80
Neuron.....	80
Neuron layer.....	81
Softmax classifier.....	81
Deep neural networks.....	82
Cost function.....	82
Training.....	82
Backpropagation.....	83
Words representation.....	83
Classic.....	83
Word Embedding.....	84
Unsupervised learning.....	84
Sentence representation.....	85

Word Hashing.....	85
Word embedding + word hashing.....	85
DSSM: Similarity Driven Sent2Vec Model.....	86
Recursive Networks.....	86
Sentence parsing.....	86
Max-Margin Framework.....	87
Compositional Vector Grammars.....	87
Language Models.....	88
N-gram language modeling.....	88
Recurrent NN for language modeling.....	88
Long-short memory cell.....	89
Neural Tensor Network.....	89

This summary hasn't been revised by any professor

If you find any errors please contact me :)

By Flavio Primo

Introduction

NLP applications

Goals:

- **analyze** and **generate** dialogues
- understand language **richness**. There's more to language than the words, things like intonation and gestures do change the meaning.

Example: learn language as a toddler: immersed in a rich language context environment. through sentiment instilled in language he can understand word meaning. Then learns pragmatic and sophisticated content.

2 trends:

- **machine can learn from lots of natural language text** available thanks to internet.
weblog analytics: data-mining of forums, websites, user groups
- **conversational agents** become important form of computer-human interaction.
much human-human communication handled by computers.

Application examples: piece of sw that requires knowledge of human language

- question answering
- summarization
- conversational agents
- machine translation

Major topics

Various levels of language that are connected together:

- **words**
- **syntax**
- **meaning**: words can have different meanings, language is ambiguous (so need specific lang to program)
- **discourse**

Can disambiguate meaning of each level because of knowledge permeating from other levels, through probabilistic means.

Ambiguity is pervasive in natural languages (because of this programming languages must be different -> unambiguous).

politeness: language rules + social rules need to be respected in order to have a meaningful conversation

Techniques

New techniques: from symbolic aspects (FLC) to probabilistic aspect (ML and deep learning approaches).

symbolic ways:

- FSA
- Context-free grammars
- Augmented grammars: unification, lambda calculus
- FOL

to more statistical approaches:

- probability models
- supervised ML
- deep learning for NLP



How it works

Put tags of different kinds on each word applied at each level to get to the final meaning.

Categories of knowledge:

- **phonology**: language's sounds
- **morphology**: how words are formed and relationships to other words
- **syntax**: arrangement of words and phrases to create well-formed sentences in a language
- **semantic**: relating to meaning in language or logic
- **prosody**: pattern of rhythm and sound
- **deviant speech**: understand from non-native speaker or person with disabilities
- **pragmatics**: rules that can be taken in account in a social environment (meaning mediated by the environment)
- **conversation**: when speaking with someone

Ambiguity

Sentences can be ambiguous because of various reasons. Example: "I made her duck"

- **lexical category**: duck → noun OR verb, her → possessive OR dative
- **lexical semantics**: make → create OR cook
- **grammar**: transitive (expect one noun direct object) OR ditransitive (expect 2 noun objects) OR action transitive (expect another verb OR noun direct object)

Solution

Attach tags to differentiate and add interpretation.

Can apply various **approaches**:

- **knowledge from other levels**: knowledge from other levels can remove ambiguity
- **pipeline**: ignore ambiguity as it occurs and relies on other levels to disambiguate
- **probability**: make most likely choice of meaning
- **nothing**: maybe ambiguity doesn't matter

Transducers (translate from one language to another) are difficult to implement due to ambiguity, so need for particular kind of algorithms:

- **dynamic programming**: ease computation of state-space search by storing intermediate

results to be reutilized.

- **state space search:** criteria to trim space and methodologies
- **classifiers:** ML based classifiers trained to make decisions on extracted features from the local context.

Language modelization (symbolic) was good but difficult to improve by a small percentage → data-driven methods through ML (probabilistic methods).

Words

Morphology

Foundation of a language, need to analyze and compose words.

Study of how words are build up from smaller meaningful units called morphomes:

- **stem:** core meaning-bearing units
- **affix:** before/after/inside a word. bits that adhere to stems to change meanings and grammatical function

example: cats → cat (stem) -s (affix)

Inflection and derivation

2 morphology mechanisms:

- **inflection:** affix that distinguishes grammatical forms of the same lexeme.
Example: different temporal times of a verb (cats inflected by cat)
- **derivation:** affix that indicates a change of grammatical category.
Example: create a name from adjective or verb (personal ADJ derives from person NOUN)

Operations on inflection/derivation:

- **generating:** need to derive/inflect a word: modify word to generate correct derivation/inflection.
inflection/derivation rules important to produce new words.
- **analyzing:** need to retrieve origin of inflection/derivation

Nouns and verbs

- **Nouns:** simple. Markers for plural and possessive.
- **Verbs:** complex. Markers appropriate to the tense of the verb.

Irregular derivations and inflections do exist.

Words and Transducers

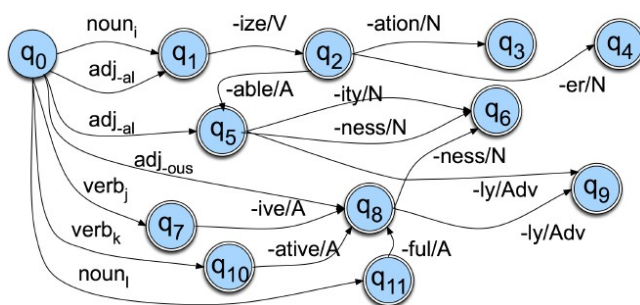
FSA good for dealing with lexicon and their inflection/derivation:

- **accept** strings that are **in the language**
- **rejects** strings that are **not in the language**

- **not necessary to list all** the language's words

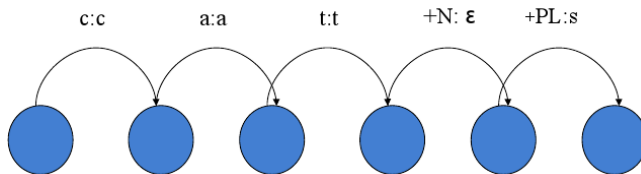
Can be represented through:

- **FSA:** cat (categorized as a regular noun) → cats (plural of regular noun adds -s); while irregular should have another pathway to get plural
example:



fossilize ⇒ *fossilization* (q_0, q_1, q_2)
equal, formal ⇒ *-ity* (q_0, q_5, q_6)
natural, casual ⇒ *-ness* (q_0, q_5, q_6)

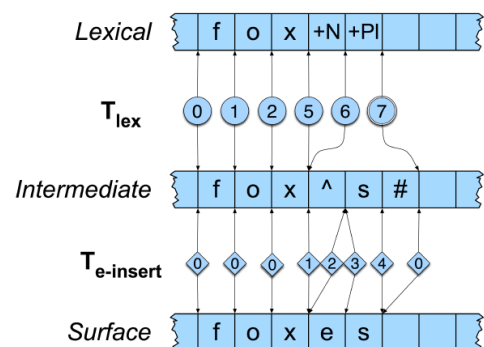
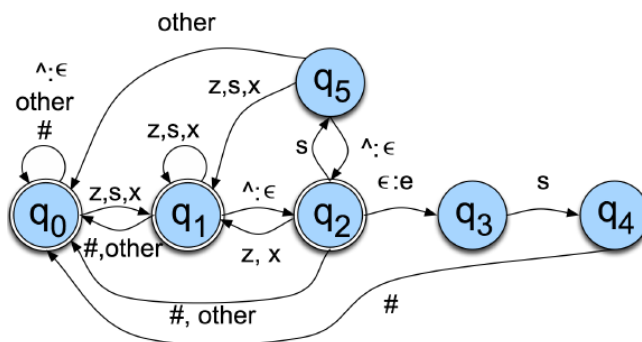
- **FST:** cat+N+PL → cats. Used for morphological analysis (spelling correction, information retrieval).



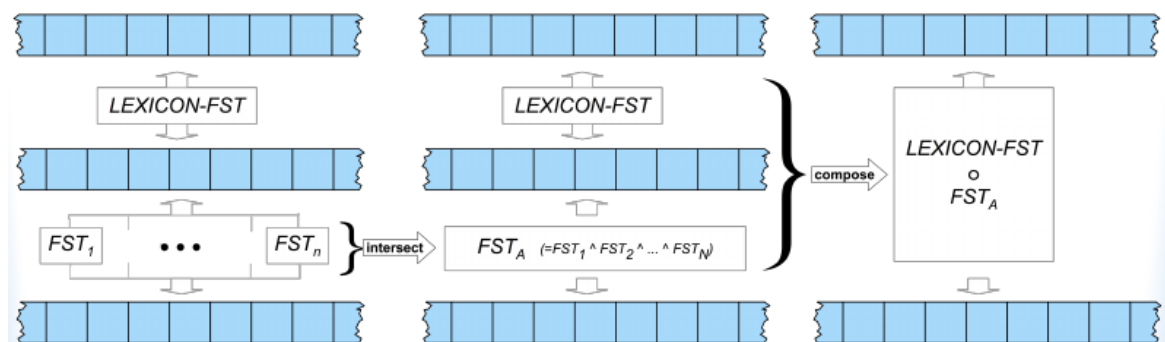
- **c:c** means read a c on one tape and write a c on the other
- **+N:ε** means read a +N symbol on one tape and write nothing on the other
- **+PL:s** means read +PL and write an s

Ambiguity: problem. Need to understand which one is the correct subpart. An intermediate unambiguous language can help the translator in the task.

The add an “e” rule as in:
 fox^s# ↔ foxes#



Cascades: can also process in multiple distinct rewrite steps. Output of one layer is the input of the next one.



Error Correction, Prediction, N-Grams

Error Correction

Available different probabilistic methods for error correction.

Bayesian model and noise channel

Goal: guess the correct “word” based on the “noisy word” (model is the “noisy channel”: as input “word” and as output the “noisy word”)

Bayesian algorithm:

Likelihood: need to estimate correct word \hat{w} based on observed sequence of characters O wrt word $w \in V$ contained in the V vocabulary.

$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w|O)$ Correct estimation \hat{w} is the most probable word $w \in V$ given a set of observed characters.

Cannot be compute directly, so we apply bayes (can remove denominator since not correlated with words):

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w|O) = \underset{w \in V}{\operatorname{argmax}} \frac{P(O|w)P(w)}{P(O)} = \underset{w \in V}{\operatorname{argmax}} P(O|w)P(w)$$

Steps:

1. **List all errors**, along with **corrections** and corrective actions

Error	Correction	Transformation			
		Correct Letter	Error Letter	Position (Letter #)	Type
acress	actress	t	–	2	deletion
acress	cress	–	a	0	insertion
acress	caress	ca	ac	0	transposition
acress	access	c	r	2	substitution
acress	across	o	e	3	substitution
acress	acres	–	2	5	insertion
acress	acres	–	2	4	insertion

2. **Normalize and Smooth** (count frequency of errors wrt corpus)

c	freq(c)	p(c)
actress	1343	.0000315
cress	0	.000000014
caress	4	.0000001
access	2280	.000058
across	8436	.00019
acres	2879	.000065

3. **Estimate** $p(t|c)$ (how many times corrective action has taken place) through a **confusion matrix**. Need to also check context (otherwise easy to apply wrong correction).

c	freq(c)	p(c)	p(t c)	p(t c)p(c)	%
actress	1343	.0000315	.000117	3.69×10^{-9}	37%
cress	0	.000000014	.00000144	2.02×10^{-14}	0%
caress	4	.0000001	.00000164	1.64×10^{-13}	0%
access	2280	.000058	.000000209	1.21×10^{-11}	0%
across	8436	.00019	.0000093	1.77×10^{-9}	18%
acres	2879	.000065	.0000321	2.09×10^{-9}	21%
acres	2879	.000065	.0000342	2.22×10^{-9}	23%

Minimum edit distance

Remove assumption of one error per word.

Consider distance between strings as a parameter related to the similarity between strings.

Distance: minimum number of editing operations needed to transform one string into another.

N-grams

Prediction of correct word based on previous words in an N sized sequence.

Even in large corpora is rare to find exact same phrase to predict, so limit N size of N-grams.

Applications

- Speech recognition
- spelling correction
- handwriting/character recognition
- machine translation

Counting

What to include and count in N-grams is application dependent: words, punctuation, plurals, articles, “uhm” (good if emotions are important), ...

Language modeling

Language model: word prediction can be modeled as $P(w_n | w_1, \dots, w_{n-1})$ (predict w_n given a previous sequence of words w_1, \dots, w_{n-1}).

Possible estimations:

- $\frac{\text{count}(w_1, \dots, w_{n-1}, w_n)}{\text{count}(w_1, \dots, w_{n-1})}$ **average** but not so good
- $P(w_1, \dots, w_n) = P(w_1) \cdot P(w_2 | w_1) \cdot P(w_3 | w_2, w_1) \dots P(w_n | w_1, \dots, w_{n-1})$ **chain rule + independence**
 - $P(A, B) = P(A|B) \cdot P(B)$ *chain rule of probability*

example:

$$\begin{aligned} P(\text{its water was so transparent}) = & \\ P(\text{its}) \cdot & \\ P(\text{water} | \text{its}) \cdot & \\ P(\text{was} | \text{its water}) \cdot & \\ P(\text{so} | \text{its water was}) \cdot & \\ P(\text{transparent} | \text{its water was so}) & \end{aligned}$$

- $P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1})$ *independence assumption (or Markov assumption)*

probability is dependent only on its immediate history.

Example:

$$\begin{aligned} P(\text{lizard} | \text{the, other, day, I, was, walking, along, and, saw, a}) = & \\ P(\text{lizard} | \text{a}) \text{ or even } P(\text{lizard} | \text{saw, a}) & \end{aligned}$$

$$\text{Example MLE bigram: } P(w_n | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

Shannon's Method

Turn N-gram prediction method to **generate** sentences like the one from which the model was derived.

It's not really generating, but more of repeating since it's entirely based on phrases on the corpus.

1. Start bigram with $(\langle s \rangle, w)$
2. keep generating (w, x) according to it's probability
3. continue till $(y, \langle /s \rangle)$ is chosen

Evaluation

Method's to measure how language models perform after training and testing.

- **Intrinsic evaluation (perplexity):** unknown words added in the open vocabulary as $\langle \text{UNK} \rangle$.

$$PP(W) = P(w_1, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, \dots, w_N)}} = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_1, \dots, w_N)}}_{\text{chain rule}}$$

nth square root to allow comparing perplexity computed with different sizes.

Minimizing perplexity is the same as maximizing the probability. Not accurate.

- **Extrinsic evaluation:** compare model A and model B performances on a real application. Good but time consuming.

Smoothing

Zero counts

Should non-existing combinations of n-grams be counted as impossible or rare?

Add bias (1) to 0 frequent n-grams

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

But results are not smoothed (normalized)

Smoothing can be done by:

- **Laplace** $P^*(w_n | w_{n-1}) = \frac{\text{count}(w_{n-1}, w_i) + 1}{\text{count}(w_{n-1}) + V}$
- **Reconstituted counts** $c^*(w_n | w_{n-1}) = \frac{[\text{count}(w_{n-1}, w_i) + 1] \times \text{count}(w_{n-1})}{\text{count}(w_{n-1}) + V}$

problem: modifies values too much

- **Good-Turing**

Josh Goodman Intuition: given probabilities of previous extracted words, how likely to

extract a new never seen word?

$$c^* = \frac{(c+1) \cdot N_{c+1}}{N_c}$$

- c^* total number of unseen words
- N_x is the number of words been extracted x times

$$P_{GT}^* = \frac{N_c}{N} \quad \rightarrow \text{discounted version}$$

other estimates are adjusted (down) to give probabilities for unseen

large c counts \rightarrow reliable, whether small c counts \rightarrow unreliable (if $c=1$ count as $c=0$)

discounting too large c not good, use threshold k whether to use c^* or not

- **Backoff:** if value not present can backoff to previous $\{n-1\}$ -gram

implementation example is the “Katz Backoff”

- **Interpolation:** combine all n-grams

adding a lambda parameter wrt to context to interpolation do embetter results

how to choose lambda? Train various $1/2/\dots$ -grams on training corpus while the lambdas are trained against a partition of the test corpus (remaining test partition used as test)

Why discounting? MLE probabilities sum = 1, but if backing off to lower model when probability is 0 would be adding probability mass \rightarrow total probability > 1

Out of Vocabulary (OOV): unigram words not seen in training, can deal in 2 ways:

- assume it can't happen
- add word token <UNK> and add it to vocabulary. Consider a threshold on rare words and substitute those rare words with <UNK> and train its probabilities like a normal word.

At decoding time use computed <UNK> probabilities for any word not in training.

Semantic

Goal: **represent meaning through a representation language.**

Semantic analysis: extract information from sentences translated in a representation language.

Lexical analysis: tries to disambiguate word meaning.

Representation language

Representation language desiderata:

- **verifiable:** provides means to compute whether sentence is true or false
- **unambiguous:** if start from word/sentence ambiguous, the result should be an unambiguous word/sentence (choose one meaning)
- **canonical form:** human language can say things in very different ways, but representation language should have same form
- **inference and variables:** should be possible to apply algorithms to representation lang. Example: evaluate truth for a question or answer to queries
- **expressiveness:** should represent every natural language expression

Language structure

All human languages are based on predicate-arguments arrangement (example: subject-verb-complements).

grammar: organizes sentences structure → representation language relies on this structure

Verb

verb defines:

- constraints on arguments → frame
- underlying semantic structure → thematic/case roles
- restrictions on arguments (semantically, example: “rock wants ...” not acceptable)

Example:

wanting[Robert] **wants** *wanted*[...]



- NP → NP
- NP → INF-NP
- NP → NP INF-VP

} Verbal frames



Syntactic analysis: analyze a string of natural language conforming to the rules of a formal grammar.

- *NP*[Robert] **wants** *NP*[Italian Food]
- *NP*[Robert] **wants** *INF-VP*[to spend less than 5 dollars]
- *NP*[Robert] **wants** *NP*[it] *INF-VP*[to be close by here]

Frames of the verb links verbal arguments (sentence structure) with the thematic/case roles (underlying semantic).

Representation language types

2 types of representation languages which focuses on:

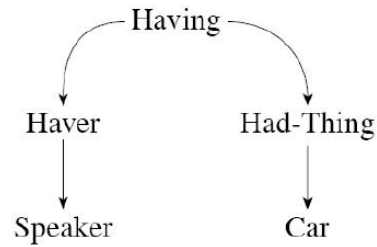
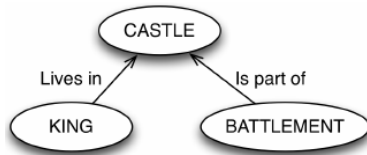
- **meaning of utterance (smallest unit of speech):** semantic networks, logics, frame-based representation
- **meaning of words:** lexical databases

Semantic Networks

Based on human semantic memory and on words only. Represented as a graph:

- **node:** words concept which coincides with words (representation is language dependent). Word meaning is given by a graph structure.
- **arc:** logical relationship among words

Networks can represent:



Domain (KB: knowledge base): defines meaning of concepts and description

Sentence: define sentence meaning (“haver” and “had-things” are relationships)

Characteristics:

- can describe **KB** and **sentences**
- KB in SN are **language dependent** → we get different structure (graphs) for the same sentence
- not a formal model (**no reasoning**)
- lots of **different implementations** (ConceptNet)
- simple but **limited**

Logic

Various types of logic: FOL (First-Order logics), DL (Description logic), ...

- formal models → **reasoning**
- **different implementations:** prolog, ontologies, production-rule systems

FOL

Representational language based on FOL.

Characteristics:

- **expressive**
- **formally defined**
- **reasoning**

Example:

“A Restaurant that serves Mexican food at Politecnico”

$\exists x \text{Restaurant}(x) \wedge \text{Serves}(x, \text{MexicanFood}) \wedge \text{Near}(\text{LocationOf}(x), \text{LocationOf}(\text{Politecnico}))$

Reasoning

FOL semidecidable: can prove if formula is true,
but if false may **not be able to prove it**



FOL restriction

(to have it decidable)

Inference: way of reasoning, most used is *modus ponens* (used as forward chaining or backward)

Example:
$$\frac{\text{VegetarianRestaurant}(\text{Rudys}) \quad \forall x(\text{VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood}))}{\text{Serves}(\text{Rudys}, \text{VegetarianFood})}$$

- *forward chaining reasoning*: proposition is a fact (added to KB), so system uses proposition in KB based on antecedents to derive new facts.

Example:

rule in KB: $\forall \text{Vegetarian Restaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood})$

when new fact ($\text{VegetarianRestaurant}(\text{Rudys})$) added to KB, then the **matching rule on the antecedent** is fired (in this case instantiate the rule with $x = \text{Rudys}$). So **consequent assert is added to the KB**: $\text{Serves}(\text{Rudys}, \text{VegetarianFood})$

- *backward reasoning*: 2 possibilities

Antecedent is true (in KB), then consequent is true, then the query is true.

Example:

fact in KB: $\text{VegetarianRestaurant}(\text{Rudys})$

rule in KB: $\forall \text{VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood})$

when query for $\text{Serves}(\text{Rudys}, \text{VegetarianFood})$ then the **matching rule on the consequent** is fired (in this case instantiate the rule with $x = \text{Rudys}$). **If antecedent is true, then the query is true.**

Query has answer if it matches on a rule's consequent and this rule's antecedent matches on the KB.

Example:

fact in KB: $\text{VegetarianRestaurant}(\text{Rudys})$

rule in KB: $\forall \text{VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood})$

when query for $\text{Serves}(x, \text{VegetarianFood})$ then the **matching rule on the consequent** is fired. So search for the antecedent in the KB $\text{VegetarianRestaurant}(x)$. **Antecedent matches** and the answer is $x = \text{Rudys}$.

forward and backward are **not complete** → **tableau calculus** (complete but time consuming)

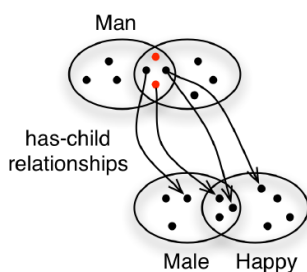
Description logic

Strange syntax based on the idea of sets.

Characteristics:

- decidable
- Subset of FOL
- Sentences and KB

KB example:



$\text{Father} = \text{Man} \sqcap \exists \text{has-child} . \text{Human}$

$\text{FatherOfHappyMaleChildren} = \text{Man} \sqcap \forall \text{has-child} . (\text{Male} \sqcap \text{Happy})$

example of some symbols:

- \sqcap intersection ($C \sqcap D \rightarrow C$ and D)
- \sqsubseteq inclusion ($C \sqsubseteq D \rightarrow$ all C are D)

Sentence example:

I have a car (definition of car owner) → $\text{CarOwner} = \text{HumanBeing} \sqcap \exists \text{owns} . \text{Car}$
 person which is human being which owns a car

Frame-based representation

Example: "I believe Mary ate British food" based on feature structure

(I → SPEAKER)

BELIEVING		
BELIEVER	SPEAKER	
BELIEVED	EATING	
	EATER	MARY
	EATEN	BRITISHFOOD

- features → slots
- values → fillers

values can be frames

Tools

Practical ways to use representation languages.

- Production-rule systems } FOL subset
- Prolog }
- ontologies, based description logic

Production-rule Systems

Based on forward chaining inference. Define facts which represent status and then apply rules using facts in the KB.

Example:

```
(def facts startup "Refrigerator Status"
  (refrigerator 1 light on)
  (refrigerator 1 door open)
  (refrigerator 2 door open))
```



Fact list

```
(refrigerator 1 light on)
(refrigerator 1 door open)
(refrigerator 2 door open)
```

$\forall x \text{ Refrigerator}(x, \text{open}) \wedge \text{Refrigerator}(x, \text{light on}) \Rightarrow$
 $\text{Refrigerator}(x, \text{food spoiled})$

```
(defrule example-rule "example"
  (refrigerator ?x light on)
  (refrigerator ?x door open)
  =>
  (assert (refrigerator ?x food spoiled)))
```



Fact list

```
(refrigerator 1 light on)
(refrigerator 1 door open)
(refrigerator 2 door open)
(refrigerator 1 food spoiled)
```

Prolog

Based on backward chaining inference.

KB

```
mortal(X) :- human(X).
human(socrates).
```

```
?- mortal(socrates).
yes
```

[:-] means [←]

$\text{Human}(\text{socrates})$

$\forall x \text{ Human}(x) \Rightarrow \text{Mortal}(x)$

$\text{Mortal}(\text{socrates})$

1. search for
"mortal(X=socrates)"
→ not found
2. match consequent
"mortal(X=socrates)"
3. search for
"human(socrates)" →
FOUND

Ontologies

Formal definitions of concepts belonging to a domain. Ontologies are language independent so we

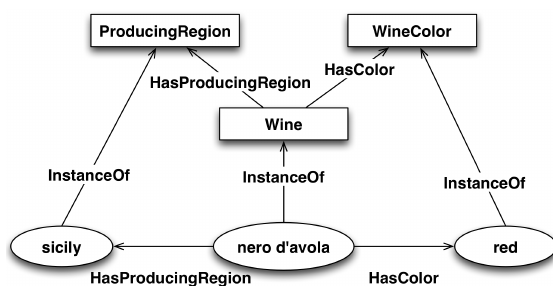
obtain same structure.

Most ontologies are composed of:

- **classes:** set of elements or a concept or a type. Examples: winery, wine
- **individuals:** element, concept, instance. Example: champagne
- **attributes:** simple properties (in OOP class fields), it has primitive data type like string. Example: price
- **relationships:** semantic characterization of a concept, relationships among classes. Examples: winery sells champagne
- **data type:** type of attribute, like string int
- **restrictions:** attribute value must >0
- **logical rules:** further specify the domains

Can think as classes as db schema and instances as records in the database.

Ontologies can represent KB and sentences (in the form of graph or list of rules):



KB (graph)

$hasParent(?x1, ?x2) \wedge hasBrother(?x2, ?x3) \rightarrow$
 $hasUncle(?x1, ?x3)$
KB (list of rules)

OWL web ontology language

- OWL Lite: taxonomies and simple constraints
- OWL DL: represents description logic
- OWL Full: higher order logics (not decidable)

Open and closed world

2 approaches for relationships:

- **closed world:** entity not found is false (neg as failure). system assumes to have complete knowledge. example: PROLOG
- **open world:** entity not found is unknown, so neither true or false, not enough info to decide. example: OWL, FOL. limits kind of inferences, but represents limited knowledge by agents.

Example: KB contains “Giovanni is an architect, g is not a physicist”

is Giovanni an engineer?

- closed world: no
- open world: unknown

is Giovanni a physicist?

- closed world: no
- open world: no

OWL

set of axioms.

language composed by:

- **class**: define concepts, can specify human readable labels for concepts in various languages
- **property**: object (relationship: subclass, disjoint, equivalent) and datatype (attribute)
- **restrictions**: type, cardinality
- **characteristics of object properties**: transitive, ...
- **individuals**

Domain is the starting point and the range is the arrival point. domain can be a class and range a property. Owl is a markup language similar to xml.

Example:

```
<owl:Class rdf:ID="Wine">
    <rdfs:subClassOf
rdf:resource="#PotableLiquid"/>
    <rdfs:label
xml:lang="en">wine</rdfs:label>
    <rdfs:label
xml:lang="fr">vin</rdfs:label>
    ...
</owl:Class>

<owl:ObjectProperty
rdf:ID="madeFromGrape">
    <rdfs:domain
rdf:resource="#Wine"/>
    <rdfs:range
rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```

SPARQL

Query language for ontologies. Language is similar to SQL and permits to traverse graphs.

- Used for:
- reasoning → get new knowledge
 - question answering

Example:

```
PREFIX food: <http://somewhere/example#> <!--prefix: specifies global
identifiers-->
SELECT ?wine
WHERE
{
    ?wine food:madeFromGrape food:ChardonnayGrape .
    ?wine food:year ?year .
    FILTER (?year > 2004)
}
```

It queries a graph, so you query for node which posses some arc type and node which have some properties (through filters).

SWRL is for reasoning and is another way to make queries.

Top and domain ontology

Ontologies describe **domains**. So **top ontologies describe basic concepts** like time and space (easier to define concepts by reusing them).

Not domain ontologies (like CYC) tries to **define all human knowledge as logical formulas** (not based on owl). Can make inferences, but ontology is so big that results are not consistent because it

is enriched by different authors.

Pros:

- abstract → search for concepts instead of words
- navigate through ontologies to gain new knowledge
- calculate similarities through distances in a graph

Tools

- **JENA**: java based lib that navigates OWL dbs and queries with SPARQL. Supports various reasoners.
- **Protege**: is an ontology editor

Semantic Analysis

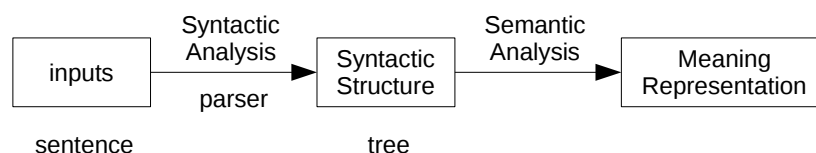
Goal of semantic analysis is, starting from a KB or a sentence in natural language, to give a meaning representation with the help of a representation language.

3 approaches to semantic analysis:

- **syntax driven**: semantic analysis: based on lexicon and grammar. Domain independent.
- **semantic grammar**: grammar elements are semantic entities. Domain dependent.
- **information extraction**: extracts small amount of information. More naive wrt the other 2

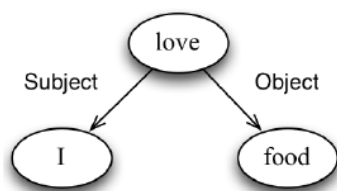
Syntax-based semantic analysis

Based on **compositionality**, meaning obtained from meaning of subparts obtained by the structure and the order of words inside of a sentence.

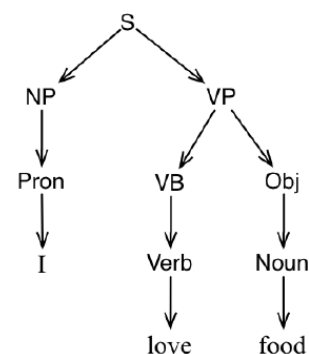


Syntactic representations

Dependency tree/graph



Parse tree



way of showing syntax of sentences as a set of subparts in a tree shape.

- S: sentence is the root
- NP: node phrase
- VP: verbal phrase
- VB: verbal
- DT: determinative articles

with a context free grammar can generate a syntactic struct tree.

Semantic augmentation

With CFG we start from leaves generating then a tree, semantic augmentation adds meaning through FOL language to each of the rules of the CFG (gives semantic to leftmost).

lambda reduction used to compute the added semantic augmentation.

Example (semantic augmentations are in brackets):

$$\begin{aligned}
 \text{ProperNoun} &\rightarrow \text{AyCaramba} && \{\text{AyCaramba}\} \\
 \text{MassNoun} &\rightarrow \text{meat} && \{\text{Meat}\} \\
 \text{NP} &\rightarrow \text{ProperNoun} && \{\text{ProperNoun.sem}\} \\
 \text{NP} &\rightarrow \text{MassNoun} && \{\text{MassNoun.sem}\} \\
 \text{Verb} &\rightarrow \text{serves} && \{\lambda x \lambda y \exists e \text{Isa}(e, \text{Serving}) \wedge \text{Server}(e, y) \wedge \text{Served}(e, x)\} \\
 \text{VP} &\rightarrow \text{Verb NP} && \{\text{Verb.sem}(\text{NP.sem})\} \\
 \text{S} &\rightarrow \text{NP VP} && \{\text{VP.sem}(\text{NP.sem})\}
 \end{aligned}$$

The problem is on how to associate to each formula the right meaning: works for very small CFG.

Idioms

Often a sentence is more of the sum of it's parts. Solve through adding dedicated rules.

Example:

“the tip of the iceberg” stands for “the beginning”

NP \rightarrow the tip of the iceberg {Beginning}

Still an open problem because so much idioms do exist.

Semantic Grammars

Syntactic grammars are not good for semantic analysis. Best to represent sentences in a more abstract way and more prone to extract meaning.

Examples:

“i want to go to eat some italian food”

- InfoRequest \rightarrow User wants to go to eat FoodType TimeExpr
- Food Type \rightarrow Nationality FoodType

“when does it arrive in Dallas?”

- InfoRequest → When does Flight arrives in City

Pros:

- very effective

Cons:

- rules design
- domain specific (difficult reuse)

Information Extraction

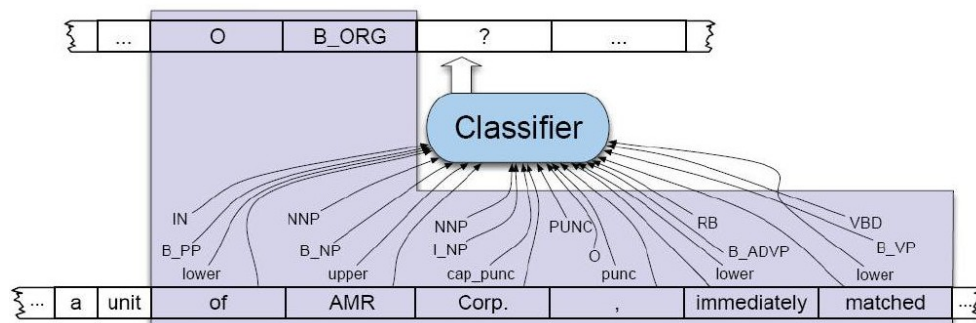
Extract particular info from a sentence (not interested in the whole sentence). Goal is to fill predefined templates.

NER Named Entity Recognition

Understand meaning related to a word given a context. Example: “Washington” can be a surname or the name of a city. Various methods:

- **REGEX** (search for urls or email addresses)
- **Gazetteers**: based on a vocabulary
- **Machine Learning**: used mainly to disambiguate.

Example: term belongs to more gazetteers and ML disambiguates among them.



ML classifies words, given current word it analyze a window of words around it to give some context.

Words	Label
American	B _{ORG}
Airlines	I _{ORG}
,	O
a	O
unit	O
of	O
AMR	B _{ORG}
Corp.	I _{ORG}
,	O
immediately	O
matched	O
the	O
move	O
,	O
spokesman	O
Tim	B _{PERS}
Wagner	I _{PERS}
said	O
.	O

Lexical semantics

Represent the meaning and relations of individual words.

Lexeme and Lexicon

Lexeme: smallest unit with

- *Orthographic* form (written form)
usually given in base form (lemma).
- *Phonological* form (spoken form)
- *Sense* (the meaning)

Lexicon: collection of lexemes (including special forms, like compound nouns)

Lexemes relationships

Polisemy/Homonymy

Polisemy

Lexeme with same form, but with **more related senses**.

Example:

- “The bank is constructed from red brick” (Bank as building)
- “I withdrew the money from the bank” (Bank as financial establishment)

Methaphor and Methonymy

- **Methaphor:** analogy between two things or ideas, the analogy is conveyed by the use of a

metaphorical word in place of some other word (example: “Germany will pull Slovenia out of its economic **slump**”)

- **Metonymy**: a concept is denoted by naming some other concept closely related to it (example: “The **White House** announced yesterday...”)

Homonymy

Different lexemes with “same” form, but with **more unrelated senses**.

Homonym can be:

- **Homographs**: same orthographic form (example: “conduct” noun and “conduct” verb).
Difficult disambiguation for TTS and information retrieval for homographs.
- **Homophones**: same phonological form (example: “piece” and “peace”).
Spelling correction (people confused by homophones) and speech recognition are affected by homophones.
- **Homograph + Homophones**: same orthographical + phonological form (example: “bank” financial establishment and “bank” the land alongside or sloping down to a river or lake).

Synonymy

Different lexemes with **same meaning** (example: automobile ↔ car)

Substitutability: sentence meaning doesn’t change if exchanging 2 lexemes (however perfect synonyms are rare).

Antonymy

Different lexemes with opposite **meaning** (example: dark ↔ light).

Hyponymy/Hypernymy

Hypernymy

Hyponym lexeme denotes a **subclass of another lexeme** (example: “dog” is hyponym of “canid”).

Hyponymy

Hypernym lexeme denotes a **superclass of another lexeme** (example: “canid” is hypernym of “dog”).

Meronymy/Holonymy

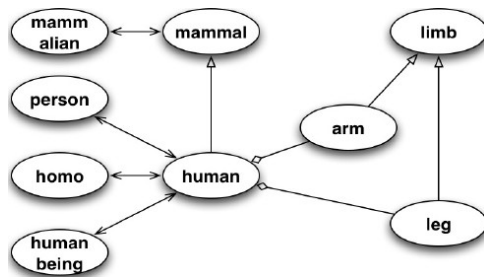
Meronymy

Meronym lexeme denotes a **constituent part of/member of another lexeme** (example: “trunk” and “limb” are meronyms of “tree”).

Holonymy

Holonym lexeme denotes the **whole of a lexeme that denotes a part of it** (example: “canid” is hypernym of “dog”).

Lexical Database



- **node:** word
- **arc:** lexical relationships
 - ◀— Hyponym / Hypernym
 - ◊— Meronym / Holonym
 - ↔— Synonym

Wordnet

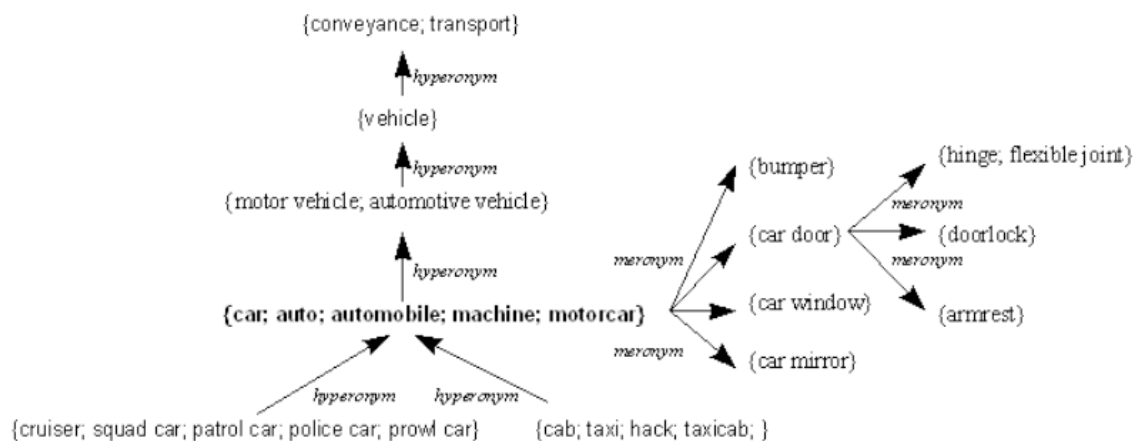
Lexical database implementation of the english language.

Synsets: way to organize lexemes in Wordnet. Every synset has a specific sense and sense description. Synsets can contain synonym and lexemes can also appear in more than one synset (homonymy/polysemy).

Relations:

- *Semantic relations:* connect synsets
- *Lexemes relations:* connect lexemes

Synset example:



Structure

- **noun/verbs:** synset's taxonomies
- **adjectives:** pairs of opposite lexemes form a group
- **adverbs:** connect related adjectives

Wordnet only contains the english open vocabulary (no pronouns, prepositions, ...).

Domains

Category labels associated to synsets (nouns and verbs) and lexemes (adjectives and adverbs).

Example: domains for "light" are "physics" and "natural philosophy"

Instance

Introduced in last Wordnet version, difficult to separate class and instances → proper nouns are instances (there may be exceptions). Usually done with ontologies.

Example: “Nero d’Avola” can be a class (set of bottles) or element of the set Wine.

Attribute

Adjectives represents values associated to a noun. Usually done with ontologies.

Example: noun “weight” has attribute adjectives “light” and “heavy”

WordNet translation

Aim to translate WordNet for many languages:

- **MultiWordNet**: one-to-one synset translation.
Not sound → because of synonyms.
Easy to add new languages but not sound
- **EuroWordNet**: each language has it’s sets of lexemes which share different meanings between words.
Sound → use of ILI intermediate language to connect lexemes of different languages.
Structure remains the same and only the language changes.
Difficult to add new languages but sound.

Lexical DB and NLP

Semantic similarity between word W_1 and word $W_2 \rightarrow$ **distance** (also weighted) in terms of relations between words (the less distance the similar).

In WordNet: $d_{SN}(W_1, W_2) = \min_{\substack{S_1 \in \text{synsetOf}(W_1) \\ S_2 \in \text{synsetOf}(W_2)}} d_{SYN}(S_1, S_2) \quad d_{SN}(S_1, S_2) = \text{min path}(S_1, S_2)$

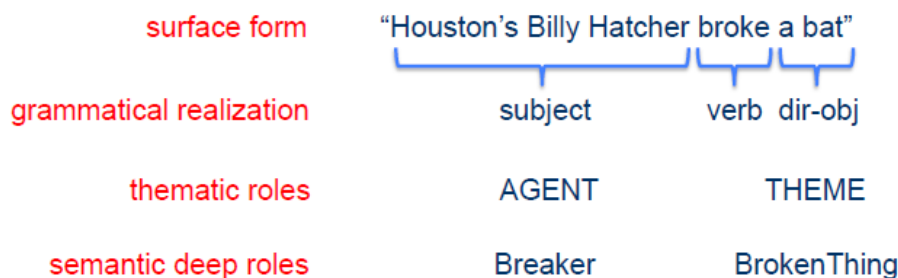
Usages:

- **Clustering**: divide similar words and/or documents in clusters based on word’s distances
- **Advanced search engines**: search for words and its properties (synonyms, ...), ...
- **Lemmatization**: get root form from flexed forms

Internal structure of words

- **Thematic roles**: roles associated with verbal arguments.
 - Agent → volitional actor (Opener, OpenedThing, Breaker, ...)
 - Theme → inanimate object affected by the action (OpenedThing, BrokenThing)
 - ...

Linking Theory: semantic roles can be seen as an intermediate level. But doesn’t consider



FrameNet: English lexicon listing syntactic and thematic combination of each word (not

only verbs).

Word → Frame (which has Frame Elements that define thematic roles) → Patterns

- **Selectional restriction:** constraints that verbs pose on their arguments.

Example: “I[AGENT] wanna eat someplace[THEME] that’s close to Politecnico”

THEME should be edible (check with Lexical Database)

- **Primitive decomposition:** decomposing words in primitive parts which carry semantic information.

Example: “The waiter brought Mary the check” → “brought”: physical movement of an object + change of possession/control of an object

- **Semantic fields:** takes into account the background information that lexemes shares (from lexical DBs)

Word Sense Disambiguation (WSD)

WSD as a side effect of semantic analysis.

Selectional Restrictions

Restrictions eliminate ill-formed components → only right meaning remains

Possible situations:

- **Unambiguous argument:** “I wash dishes” (“wash” requires something washable)
- **Ambiguous argument:** “Which airlines serve Denver?” and “Which one serve breakfast?” (“Denver” is a *location* and “breakfast” is *edible*)
- **Ambiguous argument and predicate:** “I’m looking for a restaurant that serves vegetarian dishes” (this case only one interpretation makes sense so ok, but there can be **ambiguous sentences and metaphors**)

Machine Learning

Classify words by meaning of a stochastic model (meanings → classes).

- **Input:**
 - **word** to classify
 - **text portion** where it’s embedded (context)
 - **Window** POS (part of speech) of the words (target and surrounding context, important to take right size, too small → no info and too big → too info)
example: “An electric guitar and bass player stand off to one side not really part of the scene, just as a sort of nod to gringo expectations perhaps”
- **Output:** right class

Features: input transformed into a set of features. Each word has a vector of feature pair name/value used to test and train the model.

- *target word*
- *target word collocations:* info about context words (usually in base form) and their position in the text.

Example: window=+/-2: “guitar and **bass** player stand”

[guitar, NN, and, CJC, player, NN, stand, VVB]

- *target word co-occurrences*: whether a given word (usually in base form) appears in the context of the target word or not and with what frequency.

Example: collect most frequent words

[fishing	big	sound	player	fly	rod	pound	double	runs	playing	guitar	band]
[0	0	0	1	0	0	0	0	0	0	1	0]

Supervised Machine Learning

Training: with the previously described inputs and outputs.

Training set example with format <features>;<class>: [guitar, NN, and, CJC, player, NN, stand, VVB], [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0], bass; right class: 2

Popular models: Naïve Bayes, Decision lists/trees, Neural Nets, Support Vector Machines, ...

Naïve Bayes

- **P(s)**: sense prior probability
- **v_j**: j-th feature
- **P(v_j | s)**: probability of feature v_j, given sense s

$$\hat{s} = \underset{s \in S}{\operatorname{argmax}} P(s | v_1, \dots, v_n) = \underset{s \in S}{\operatorname{argmax}} \frac{P(v_1, \dots, v_n | s) \cdot P(s)}{\underbrace{P(v_1, \dots, v_n)}_{\text{bayes}}} = \underset{s \in S}{\operatorname{argmax}} \underbrace{P(v_1, \dots, v_n | s) \cdot P(s)}_{\text{denominator does not depend on } s \rightarrow \text{remove}} =$$

$$\underset{s \in S}{\operatorname{argmax}} P(s) \prod_{n=1}^j P(v_j | s)$$

Bootstrapping:

1. **seed**: start with a small number of instances of each sense and lexeme.

Can be chosen at random or “one per sense collocation” (for each lexeme discover words that co-occur frequently and use sentences where this words appear as seed for the target lexeme, always with same sense)

2. **train** classifier on the seed
3. **label** a larger set of words with the trained classifier
4. **check correctness** of the labeling
5. REPEAT

POS tagging, syntactic analysis

Part of Speech tagging

Part of Speech

- N - Noun
 - V - Verb
 - ADJ - Adjective
 - P - Preposition
 - ADV - Adverb
 - Article
 - Interjection
 - PRO - Pronoun
 - Conjunctions
- } POS
Part of Speech

POS can be more fine grained or more coarse grained and pertitioned in different ways.

Open and Close classes

- **Closed:** function words which play a role in grammar and not added frequently
- **Open (N, V, ADJ, ADV):** words which are added all the time

POS tagging

POS tagging: assign to each word a POS. Example: Koala → N

First step of various activities: speech synthesis, parsing (understand and give meaning to subparts), information exctraction, machine translation.

Difficulty in tagging: choosing the right tag for a word (ambiguity)

Steps for POS tagging:

1. **choose a standard set of POS tags** to work with
2. **choose POS method** for tagging
 1. *Rule-based* (ENGTWOL)
 1. assign all possible tag to each word
 2. remove all not possible combinations of tags
 2. *Stochastic* (HMM, MEMM)
with HMM:
 1. model with the Hidden Markov chain
 2. decode (Viterbi)

3. **Evaluation** of the chosen tagging method

error rates on some tags/words wrt to a gold standard (perfection impossible), tag confusions

Error analysis through confusion matrix:

	IN	JJ	NN	NNP	RB
IN	—	.2			.7
JJ	.2	—	3.3	2.1	1.7
NN		8.7	—		

how many times IN was the correct tag in place of JJ

HMM POS tagging

HMM can be considered as moving on 2 levels: **lower** level is known and it's the **sequence of observations (words)**, while the **upper** level (where we are moving) it's unknown and it's the **sequence of corresponding tags**.

Corresponding tags are unknown, so consider all possible sequence of tags and choose the most probable tag sequence.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)} = \underset{t_1^n}{\operatorname{argmax}} \underbrace{P(w_1^n | t_1^n)}_{\text{likelihood}} \underbrace{P(t_1^n)}_{\text{prior}}$$

Want single tag sequence t_1^n (tags from 1 to n) which has the highest probability (can compute with **Bayes rule** as shown).

Probabilities types (encoded in two matrices):

- **tag transition probabilities** (matrix A, upper level transitions: tags)

$$P(t_i | t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

- **word likelihood probabilities** (Matrix B, lower level transitions: words)

$$P(w_i | t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

Markov chain

Weighted finite-state automaton (WFST): FST with probabilities added to arcs

Markov chain: special WFST where the input sequence which state the automaton will go through uniquely (can't represent ambiguous problems). One observable level.

HMM chain: extension of Markov Chain where inputs are different from states (don't know which state we are in, can represent ambiguous problems). 1 observable level (input) and 1 hidden level.

- States: $Q = q_1, \dots, q_N$
- Observations: $O = o_1, \dots, o_N$
- Transition probabilities: $A = \{a_{ij}\}$ $a_{ij} = P(q_t = j | q_{t-1} = i)$ with $1 \leq i, j \leq N$
- Observation likelihoods: $B = \{b_i(k)\}$ $b_i(k) = P(X_t = o_k | q_t = i)$ with $1 \leq i \leq N$
- Initial probability vector: $\pi = P(q_1 = i)$ with $1 \leq i \leq N$ (beginning and ending conditions)

Viterbi algorithm

Dynamical programming algorithm that allows to compute the most probable path. In this case applied to HMM graphs to find most probable tag sequence.

Formal Grammars

Grammar (or Syntax)

Studies the way words are arranged together

Constituency

Constituent: groups of words may behave as a single unit or phrase.

Example: noun phrase (as: Michael, she, Russian Hill, ...) usually precedes verbs.

Various competing theories on how to form and understand how constituents are combined.

Constituents grouped into classes have similar:

- internal structure
- external behaviour (example: noun phrases come before verbs)

Context Free Grammars (CFG)

Consists of:

- $R: A \rightarrow \beta$ **rules:** equations that consist of a single non-terminal on the left and any number of terminals on the right.

Example: some rules to determine noun phrases

$$\begin{aligned} NP &\rightarrow Det\ Nominal \\ NP &\rightarrow ProperNoun \\ Nominal &\rightarrow Noun | Nominal\ Noun \end{aligned}$$

Generativity: FSA and FST can represent those rules in a generative or analytical way.

Derivation: sequence of rules applied to a string that accounts for that string.

- Σ **terminals:** words
- N **non-terminals:** constituents

Parsing

Rules can be used to either **analyze** or **synthesize** language strings.

Derivations: sequence of rules applied to a string that accounts for that string.

Parsing: Process of taking a string+grammar \rightarrow one/more parse trees for that string (same as running a FST transducer with a tape).

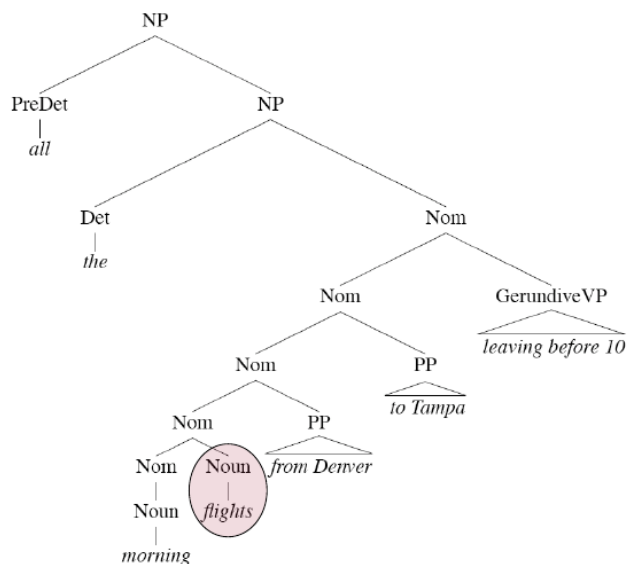
English Grammar Example

Sentences

Sentences can be of various types depending on the intention and purpose:

- **Declaratives:** $S \rightarrow NP\ VP$
- **Yes-No questions:** $S \rightarrow Aux\ NP\ VP$
- **Imperatives:** $S \rightarrow VP$
- **WH questions (when, who, ...):** $S \rightarrow WH-NP\ Aux\ NP\ VP$

Noun phrases



Noun phrases can be very complex even with simple rules.

NP → Det Nominal

Example: “All the morning flights from Denver to Tampa leaving before 10”.

HEAD: highlighted in the diagram is the most important word of the sentence. There exist various algorithms that to find it. Position depends also on the language, mostly at beginning or at the end.

Determiners: noun phrases can start with Det which are:

- simple lexical items: the, this, a, an, ...
 - possessives: John’s car
- or recursive versions of the previous:
John’s sister’s husband’s son’s car

Nominals: contains head and any pre- and post- modifiers of the head.

Pre-

- Quantifiers, cardinals, ordinals (example: three cars)
 - Adjectives and appositions (example: large cars)
- Ordering constraints:
Three large ~~three~~ cars

Post-

- Prepositional phrases (example: from Seattle)
- Non-finite clauses (example: arriving before noon)
- Relative clauses (example: that serve breakfast)

Agreement: disambiguate references, helps to connect subject to verb. Example: Det and HEAD nouns needs to agree on the number.

Verb phrases

Starts with an head verb followed by various arguments, described by VP rules.

Subcategorization: to support all verbs need a large pool of rules → subcategorize verbs in a language according to the sets of VP rules that they participate in.

Need constraints to limit overgeneration.

Problems:

- **Overgeneration:** rules that can generate correct sentences as well as incorrect ones not respecting agreements.

Weak solution: add non-terminals → doesn’t scale, too much rules

Solution: probabilistic approaches

Probabilistic approaches

Treebanks

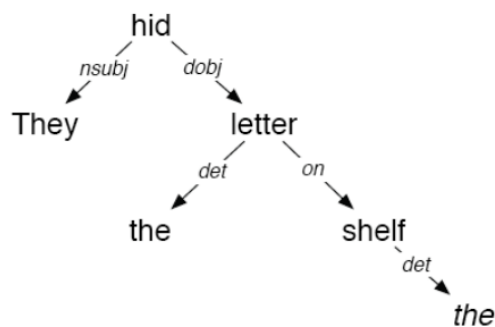
Specific corpora which includes a large quantity of documents already syntactically analyzed (all sentences have correct parse string).

Treebank grammars: treebanks implicitly define a grammar rule set. Also useful to count how probable are some expansions wrt to others.

Treebank grammars avoid recursion and tend to have flat grammars.

Heads: can be found by using tree traversal rules specific to each non-terminal in the grammar.

Dependency grammars



Highlights relationships in terms of roles between words. Better alternative than constituent “concepts”.

Dependency parsing:

- deals well with free word order languages
- faster parsing than CFG
- often capture syntactic relations

2 types: optimization-based (search a space of trees for the tree that **best** matches some criteria), Shift-reduce (a more greedy approach)

Grammars and Parsing

Shallow parsing

Analysis done with very simple non terminals (example: Company name, Verb group, Noun group, ...).

FSA sufficient to parse and it's important to avoid recursion.

Recursive Transition Networks (RTN): not a simple transition, but uses subroutines to deal with non terminals returning the result only after been parsed.

Probabilistic parsing

Add probabilities to grammars which the parser will interpret while traversing the tree.

Probabilistic CFG: find tree which have maximum probabilities.

Adds probabilities to grammar rules:

$VP \rightarrow Verb$.55	$= P(Verb \mid VP)$
$VP \rightarrow Verb NP$.40	$= P(Verb NP \mid VP)$
$VP \rightarrow Verb NP NP$.05	$= P(Verb NP NP \mid VP)$

From treebanks can be derived CFG/PCFG:

- **CFG:** collect the set of productions used in the treebank
- **PCFG:** add probabilities to the collected productions

Assumes that non-terminals expansions are independent from each other, but this leads to:

- *Structural dependencies*: same terms with different roles may need to be treated differently.

Solution “Parent annotation”: helps disambiguate how tree should be parsed based on parent node (but increase grammar size and reduces training corpus).

$VP \rightarrow VDB\ NP$

Example: instead we can write:

$\underbrace{VP \wedge S}_{\text{apply iff parent of VP is S}} \rightarrow VDB \quad \underbrace{NP \wedge VP}_{\text{apply iff parent of NP is VP}}$

- *Lexical dependencies*: some terms should be treated differently depending on the context
example: coordination problem “(dogs in houses) and (cats)” or “(dogs) in (houses and cats)”

Solution “lexical head annotation”: for each non terminal rule annotate POS+word (the most important one). Need a huge corpus to train for recognizing the lexical head.

example: E.g.: $VP(\text{dumped}, VDB) \rightarrow VDB(\text{dumped}, VDB) NP(\text{sacks}, NNS) PP(\text{into}, P)$

$$p = \frac{C(VP(\text{dumped}, VDB) \rightarrow VDB(\text{dumped}, VDB) NP(\text{sacks}, NNS) PP(\text{into}, P))}{C(VP(\text{dumped}, VDB))}$$

Probabilistic parsing

(achievable through dynamic programming) needs:

- grammar
- parser
- dictionary with POS

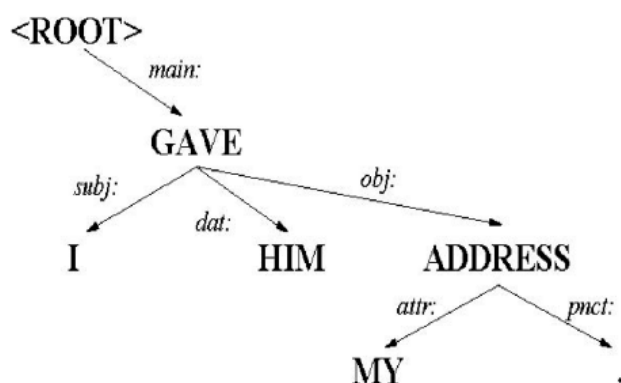
Dependency Grammars

Describes grammatical relationship between 2 words only (instead of describing parts of sentence).

Similar to “analisi logica”, search for: subject, direct object, complement.

Useful for languages where words can appear in various positions.

Parsing



Sentence parsed as a graph:

- node: words
- arcs: relationships

usually to simplify: transform graph into a tree

Possible list of dependencies:

- Subj
- Obj
- Dat
- Pcomp
- Comp
- Tmp
- Loc
- Attr
- Mod

Syntax: tools (parsing, corpora)

Parsing

Different approaches are available for parsing which leverage different language models:

- **Classical** → CFG
- **Stochastic** → PCFG, L-PCFG
- **Others** → Dependency grammars, Feature-based grammars

Classical CFG

Example: S → Det N
 S → N
 Det → the | a
 N → dog | cat

Various algorithms are available: top-down, bottom-up, earley.

Can be subject to ambiguities (can derive multiple trees).

Stochastic

PCFG

Example: S → Det N [0.8]
 S → N [0.2]
 Det → the [0.6] | a [0.4]
 N → dog [0.5] | cat [0.5]

structure and probability, can be learned from a corpus (a treebank).

Ambiguous as CFG, but trees are ranked wrt to probability.

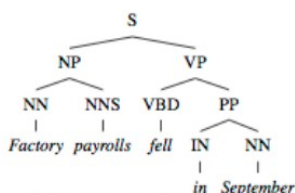
L-PCFG

Example: 1. VP(dumped) → VBD(dumped) NP(cats) PP(into) [p=3×10⁻¹¹] Lexicalized PCFG adds lexicon to PCFG.
 2. VP(dumped) → VBD(dumped) NP(sacks) PP(into) [p=3×10⁻¹⁰] Need to be trained.
 3. VP(dumped) → VBD(dumped) NP(sacks) PP(above) [p=3×10⁻¹²]

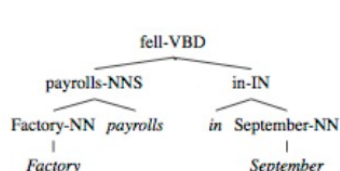
Various implementations:

- **Stanford parser:** split tree in 2 parts (words dependencies and unlexicalized tree) and train them separately.

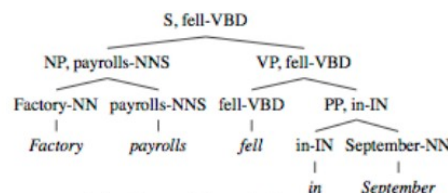
$$p(T, D) = \underbrace{p(T)}_{\text{prob of a given unlexicalized parse tree}} \cdot \underbrace{p(D)}_{\text{probability of a given dependency tree}}$$



(a) PCFG Structure



(b) Dependency Structure



(c) Combined Structure

a) retain only words and discard tags and build tree of words and corresponding post tags

b) from b can create a new grammar using only words and not non terminals (doesn't take into account meaning of where words do appear), doesn't matter for example that root is the root of the tree.

applied each time the tree is found.

Others

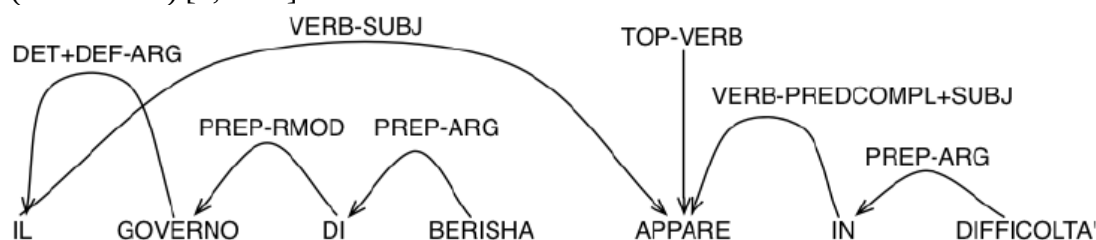
Dependency grammar

TUT

TUT treebank contains DG-tagged sentences.

Syntax: ($\langle \text{WORD} \rangle \langle \text{gender/verb conjugation/type/...} \rangle$) [$\langle \text{REF NUMBER} \rangle$, $\langle \text{RELATIONSHIP} \rangle$]

1. **Il** (IL ART DEF M SING) [5;VERB-SUBJ]
2. **Governo** (GOVERNO NOUN COMMON M SING) [1;DET+DEF-ARG]
3. **di** (DI PREP MONO) [2;PREP-RMOD]
4. **Berisha** (BERISHA NOUN PROPER) [3;PREP-ARG]
5. **appare** (APPARIRE VERB MAIN IND PRES INTRANS 3 SING) [0;TOP-VERB]
6. **in** (IN PREP MONO) [5;VERB-PREDCOMPL+SUBJ]
7. **difficoltà** (DIFFICOLTÀ NOUN COMMON F ALLVAL) [6;PREP-ARG]
8. **(#. PUNCT)** [5;END]



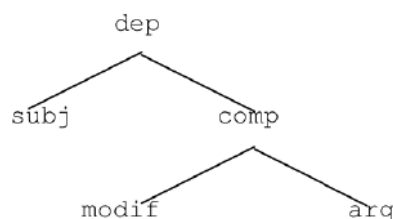
IDEAL

Dependency parser for Italian language. It outputs grammatical relationships between head word and a dependent word. Based on rules.

Rules example:

```
sogg (visitare, presidente)
comp (presidente,
repubblica.<intro=di>)
ogg (visitare, capitale)
```

Relationships organized as hierarchies:



Corpora

Collection of texts tagged by humans setting the "gold standard". Used to train statistical models.

Among the most famous corpora: Brown corpus, Penn Treebank, CoNLL

How corpora are built

Bootstrap

1. Manually tag corpus subset
2. Train model
3. Tag larger corpus subset with trained model
4. Manually revise and fix tags

Kappa measure

Measure of agreement among manually added tags: $K = \frac{P(A) - P(E)}{1 - P(E)}$

where:

- $P(A)$ observed agreement among raters
- $P(E)$ expected agreement (better if computed as free-marginal: raters don't know a priori the quantities of cases that should be distributed into each category)

Agreements can be:

- 1 → agree
- 0 → equal to chance
- -1 → disagree

Syntax: useful stochastic models

Stochastic models are used for classification (POS tagging, parsing, ...).

Supervised models are trained with a tagged corpus which output the trained language model.

Can be:

- **Generative:** Naive Bayes, HMM
- **Discriminative:** MaxEnt, MEMM, CRF

Generative

HMM (look before)

Discriminative

MaxEnt

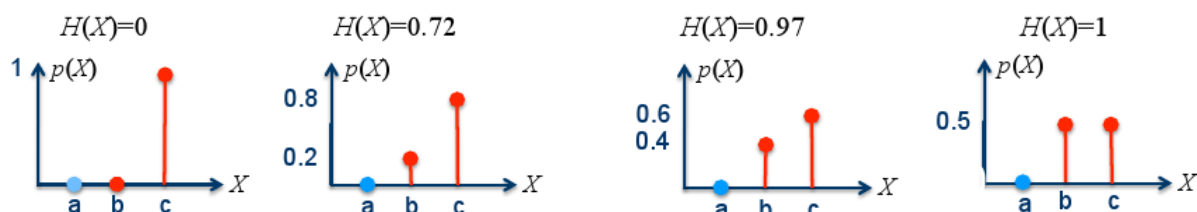
If probability distributions over partial knowledge → statistical inference: maximum entropy estimate (least biased estimate of information).

Self information of a value x of a stochastic variable X : $\underbrace{I(X=x)}_{\text{surprise to get } X=x} = - \underbrace{\log_2}_{\text{measure in bits if small} \rightarrow \text{high surprise}} \underbrace{p(X=x)}_{\text{measure in bits if small} \rightarrow \text{high surprise}}$

Entropy

measures the uncertainty of a distribution $H(X) = \underbrace{E[I(X)]}_{\text{expected surprise over } p(x)} = \sum_x p(X=x) \cdot I(X=x)$

examples:



Example: tag word “fish”

1. all possible tags (no assumptions), distribute equally

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	...
$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$...

2. From tagged corpus take only admissible POS and distribute equally

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	...
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0	...

3. From corpus 8 times out of 10 fish is tagged as noun (either NN or NNS)

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	...
$\frac{4}{10}$	$\frac{1}{10}$	$\frac{4}{10}$	$\frac{1}{10}$	0	0	0	0	0	...

4. From corpus verbs (VB) occurs as 1 out of 20

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	...
$\frac{4}{10}$	$\frac{3}{20}$	$\frac{4}{10}$	$\frac{1}{20}$	0	0	0	0	0	...

5. Then compute MaxEnt (remember to apply only certain constraints to not have it biased)

$$\hat{h} = \underset{H}{\operatorname{argmax}} p(H, O_{1:k}) = \underset{H}{\operatorname{argmax}} \exp \left(\sum_i^m \lambda_i \cdot f_i(H, O_{1:k}) \right)$$

with:

$$p(H, O_{1:k}) = \frac{\exp \left(\sum_i^m \lambda_i \cdot f_i(H, O_{1:k}) \right)}{\sum_k \exp \left(\sum_i^m \lambda_i \cdot f_i(H = h_k, O_{1:k}) \right)}$$

- λ_i weight associated to feature i
- feature i on H and O:

$$f_i(H, O_{1:k}) = \begin{cases} 1; & \text{a pattern on } H \text{ and } O_{1:k} \\ 0; & \text{otherwise} \end{cases}$$

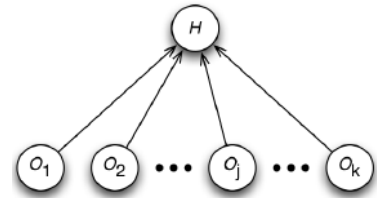
example:

$$f_i(H, O_{1:k}) = \begin{cases} 1; & H = \text{"NN"} \wedge O_1 = \text{"talk"} \wedge O_2 = \text{"is"} \\ 0; & \text{otherwise} \end{cases}$$

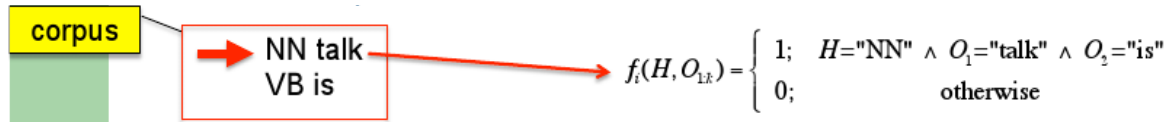
f_i is learned from a tagged corpus:

1. define template T_f

$$T_f(H, O_{1:k}) = \begin{cases} 1; & H = \text{current_hidden_state} \wedge O_1 = \text{current_word} \wedge O_2 = \text{next_word} \\ 0; & \text{otherwise} \end{cases}$$



2. Find template instance



3. select most relevant instances (occur more than v times)

expected values for f_i are then calculated from:

$$E_p[f_i] = E_{p^*}[f_i]; \forall i \quad \text{count}_p(f_i) = \text{count}_{p^*}(f_i); \forall i$$

$$\text{count}_{p^*}(f_i) = N \cdot \sum_{\{(H, O_{1:k})\}} p^*(H, O_{1:k}) \cdot f_i(H, O_{1:k})$$

Predicted by the model
Actual (from training corpus)

Pros:

- no feature independence assumption
- training algorithm guarantees convergence
- features can predicate on every piece of information
- works on sequences

cons:

- training must take into account all features at the same time

MaxEnt Markov Models (MEMM)

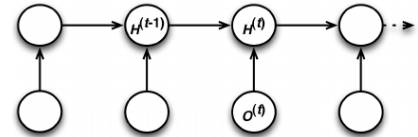
$$\hat{h}^{1:n} = \underset{H^{1:n}}{\operatorname{argmax}} p(H^{1:n}, O^{1:n}) =$$

$$\underset{H^{1:n}}{\operatorname{argmax}} \frac{\prod_t \exp\left(\sum_i \lambda_i \cdot f_i(H^t, H^{t-1}, O^t)\right)}{P(H^{t-1})}$$

with:

$$p(H^t | H^{t-1}, O^t) = \frac{\exp\left(\sum_i \lambda_i \cdot f_i(H^t, H^{t-1}, O^t)\right)}{\sum_k \exp\left(\sum_i \lambda_i \cdot f_i(H^t = h_k, H^{t-1}, O^t)\right)}$$

$$p(H^{1:n}, O^{1:n}) = \prod_t p(H^t | H^{t-1}, O^t)$$



Pros & cons same as MaxEnt, also model is bigger and slower than HMM

CRF

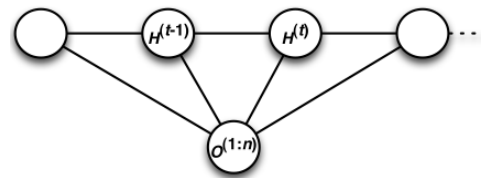
Want to predict a large number of variables that depend on each other as well as on other observed variables. CRF is a way of combining:

- **classification:** leverage a large number of input features for prediction
- **graphical modeling:** compactly model multivariate data

$$\hat{h} = \underset{H^{1:n}}{\operatorname{argmax}} p(H^{1:n}, O^{1:n}) = \underset{H^{1:n}}{\operatorname{argmax}} \exp \left(\sum_i^{m_1+m_2} \lambda_i \cdot \sum_t^n f_i(H^t, H^{t-1}, O^{1:n}, t) \right)$$

with:

$$p(H^{1:n}, O^{1:n}) = \frac{\exp \left(\sum_i^{m_1+m_2} \lambda_i \cdot \sum_t^n f_i(H^t, H^{t-1}, O^{1:n}, t) \right)}{P(O^{1:n})}$$



Pros & cons same as MaxEnt, also model is bigger and slower than MEMM

Generative vs discriminative

Generative	Discriminative
Independency assumption	No independence assumption
	Function to represent big distribution
Train small distributions	Train function parameters

Training and Cross-validation

Repeated sub-sampling validation:

1. randomly partition samples into training set and test set
2. train+test model
3. repeat at will

Model evaluation

- Confusion matrices

		PREDICTED CLASSES (e.g., TAGS)						
		IN	JJ	NN	NNP	RB	VDB	VBN
CORRECT CLASSES (e.g., TAGS)	IN	760	20	0	0	70	0	0
	JJ	20	4350	330	210	170	20	270
	NN	0	870	5460	0	0	0	20
	NNP	20	330	410	3508	20	0	0
	RB	220	200	50	0	2358	0	0
	VDB	0	30	50	0	0	1480	440
	VBN	0	280	0	0	0	260	1650

- Indexes (Precision, Recall, Mean, Weighted mean, Accuracy)

POS Tagging and Chunking

POS tagging

POS taggers do disambiguate using the context leveraging a **language model** (there can be multiple per language).

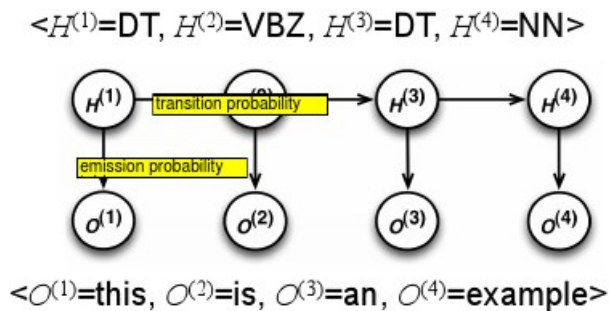
Stochastic approaches

HMM

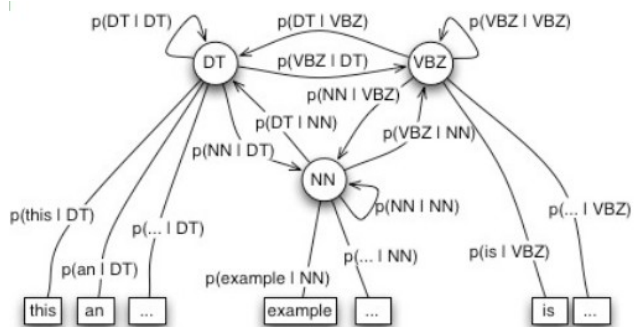
Considers a sequence at a time. Through Viterbi: given a sequence of word $\langle w^i \rangle \rightarrow$ get most probable sequence of tags $\langle t^i \rangle$.

- $H = \{t_a, t_b, \dots\}$ stochastic variables, the tag set (example: DT, VBZ, ...)
- $O = \{w_a, w_b, \dots\}$ observable variables, the word set

Unrolled view



Graph view (explode transitions)



Example: $p(VBZ|NN)$ is the probability of going from NN to VBZ (VBZ given NN)

Tools:

- **Freeling**

morphologic analyzer

lists all possible meanings

This	is	a	,	quite	simple	,	example
this	be	1	,	quite	simple	,	example
DT	VBZ	Z	Fc	RB	JJ	Fc	NN
0.999824	1	0.999969	1	0.935714	0.864583	1	1
this		a		quite	simple		
PRP		DT		PDT	NN		
0.0001755		1.01887e-05		0.0642857	0.135417		
		a					
		NN					
		1.01887e-05					
		a					
		NNS					
		1.01887e-05					

POS tagging (HMM)

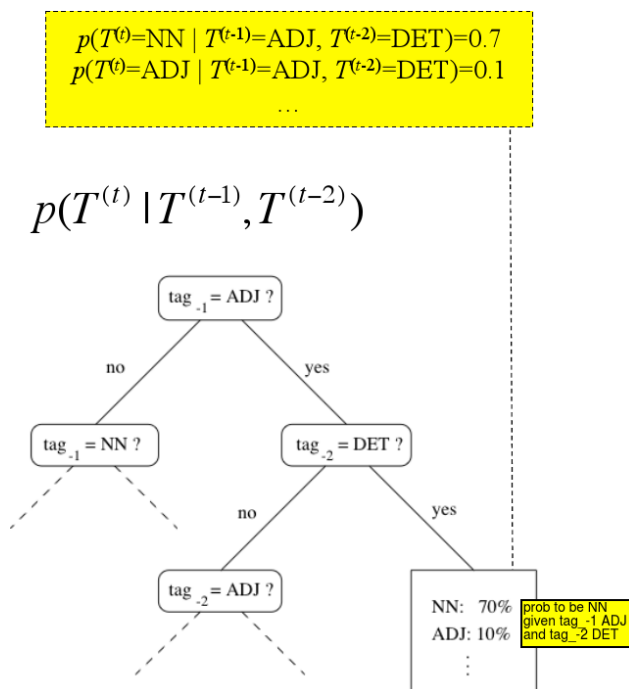
This	is	a	,	quite	simple	,	example
this	be	1	,	quite	simple	,	example
DT	VBZ	Z	Fc	RB	JJ	Fc	NN

- **Stanford POS tagger**

POS tagging (Entropy Maximization)

- **Tree Tagger**

POS tagging based on a 2^o order HMM (estimates transition probabilities by means of a model)



Chuncking (shallow parsing)

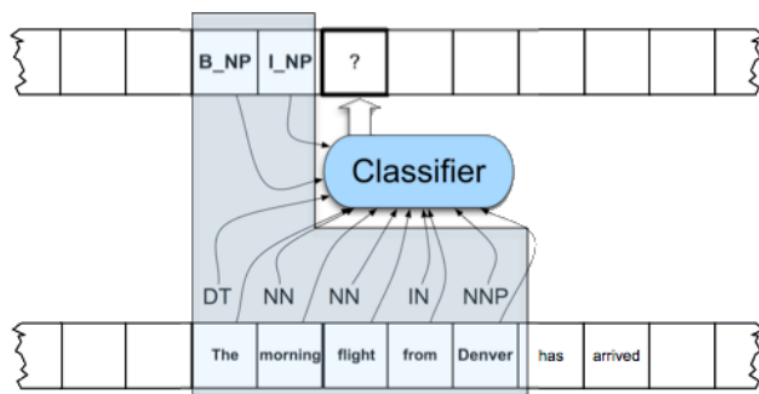
Instead of creating a tree structure, just divide the sentence in 2 chunks (easier, not recursive), leveraging POS tagging. ML approach is presented.

Training: corpus that associates words to tags.

Example:

Sentence <WORD>	Tag <TAG>	Correct tag to learn <BOUNDARIES>_<TAG> <BOUNDARIES>= B_ - begin of a chunk I_ - inside the chunk O_ - not part of a chunk
He	PRP	B_NP

Chunker slides a context window over the sentence classifying words as it proceeds.



Example: here classifier tries to label flights.

Voice

Voice adds meaning to sentences, and always adds something more (irony, tone, accent, ...).

Interested in:

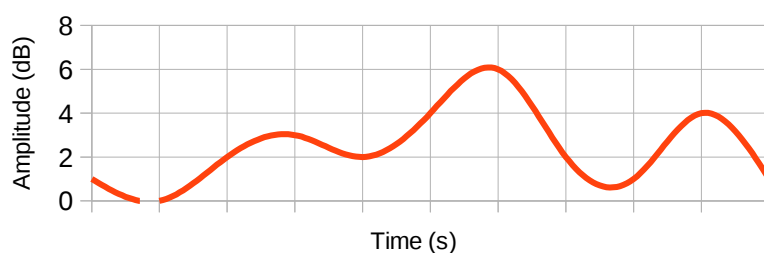
- **“Phonology”** how phonemes are placed in the sentence structure to gain meaning
- **“Prosody”** intonation, rhythm tempo, loudness and pauses and the interactions with syntax, lexical meaning and segmental phonology in spoken texts.

Problems: ASR misses prosodic details, limited knowledge of the discourse analysts.

Speech Analysis

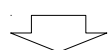
Human Voice: prosody and voice descriptors in DA

Time and frequency domain

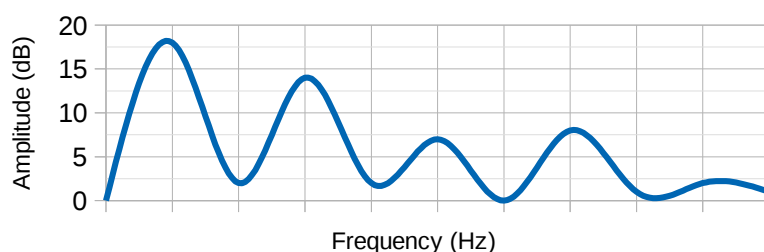


Time domain

How a signal changes over time.



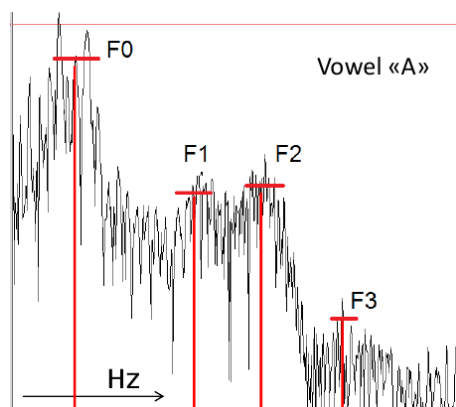
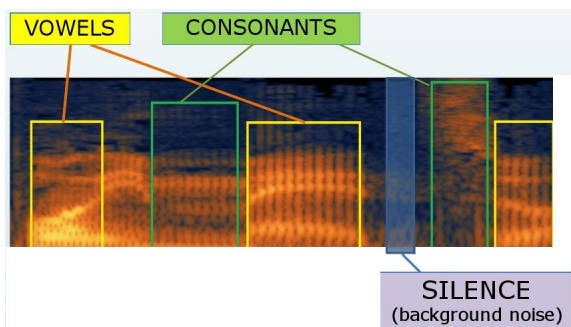
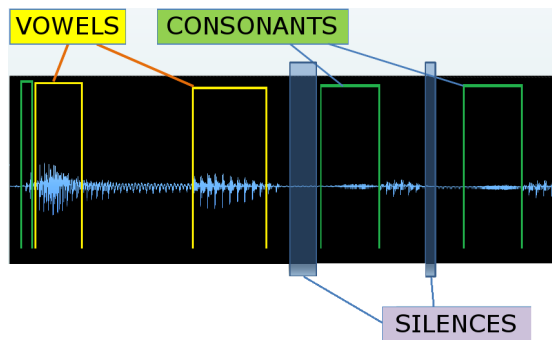
Fourier Transform: transform aperiodic signal in time domain to frequency domain



Frequency domain

How much of the signal lies within each given frequency band over a range of frequencies.

Vowels and consonants



Time domain

Frequency domain

- vowels: frequency band concentration
- consonants: noisy component

Vowel components:

- **F0: pitch**

Allows sound ordering on a frequency-related scale.

Highest spectral peaks in the sound spectrum of the voice.

Pitch can be determined only in sounds that have a frequency that is clear and stable enough to distinguish from noise.

- **F1, F2, F3: formants**

Spectral peaks in the sound spectrum of the voice.

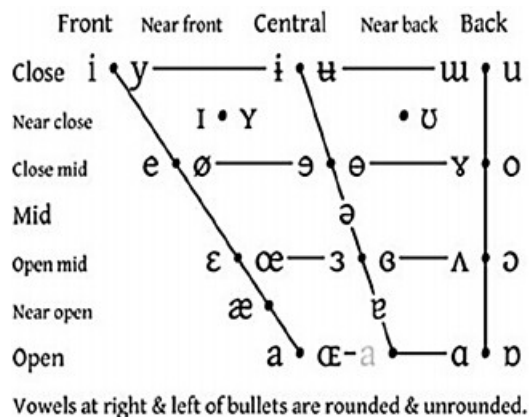
Formants create the timbre and let distinguish on the type of voice (or instrument if produced by a musical instrument).

Music sounds nice because formants have rules.

- **Bi: relative bandwidth**

Bandwidth of the spectrum.

Vowels

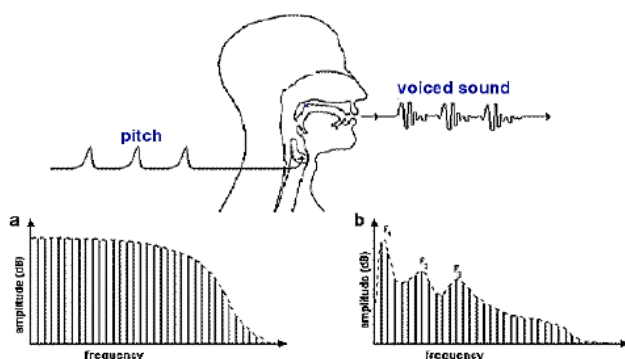


IPA (International Phonetic Alphabet):

standard representation of the sounds of oral language based primarily on the Latin alphabet

IPA vowel diagram: maps the vowels according to the position of the tongue.

- Horizontal shift: tongue position
- Vertical shift: mouth opening



Source Filter Model: models speech as a combination of a sound source:

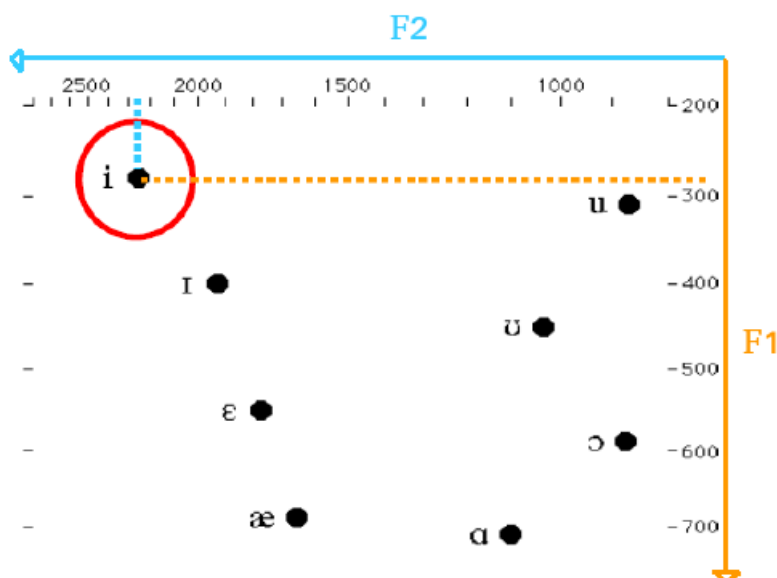
- *vocal cords*: sound source
- *vocal tract*: linear acoustic filter

Complex wave = vocal cords + vocal tract

independence of source and filter.

Problem: univocal audio characterization of vowels

F1 F2 space



Formants F1 and F2 defines a relation that describes vowel's height and frontness.

- High F2 → front vowel
- High F1 → open vowel

Bark scale

Psychoacoustic scale that ranges from 1 to 24 and corresponds to the first 24 critical bands of hearing.

Formants (in Bark):

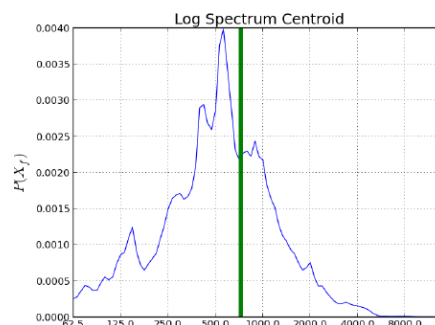
- **(F1 – F0) low tone** <3,5 – high tone > 3,5 («o» or «i»)
- **(F2 – F1) back vowel** <3,5 – front vowel >3,5 («a» or «i»)
- **(F3 – F2) central or back vowels** («e» or «a»)
- **(F2 + F1) rounded or unrounded** (example: «o» or «i»)

Features (in Hz):

- **Pitch (F0) gender, age and emotional tone** of the speaker
- **F1 opening degree of the mouth**

Other spectral features:

- **Vocal stability:**
 - *Jitter*: Frequency of a speaker's voice varies from one cycle to the next. High variability → rhythm changes
 - *Shimmer*: Same as frequency perturbation, but analogous to amplitude. High shimmer → hoarseness
- **Spectral centroid (brightness of the sound)**: spectrum center of mass (in green)



Vowels normalization

Process that model the cognitive processes that allow human listeners to normalise:

- **eliminates variations from physiological differences** of the speakers
- **preserves sociolinguistic/dialectal/cross-linguistic differences**
- **preserves phonological distinctions**

Data normalized through modified Watt-Fabricsius method (mW+F): $S(F_i) = \frac{F_i[i] + F_i[a] + F_i[u']}{3}$

Consonants

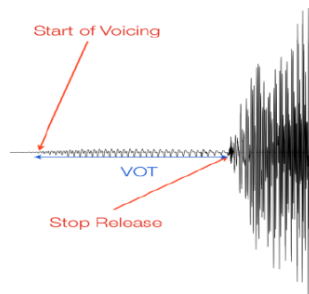
Modulation of noise for verbal communication. Can be voiced (like D) or unvoiced (like T).

Difficult to discriminate between consonants because there are no harmonic features.

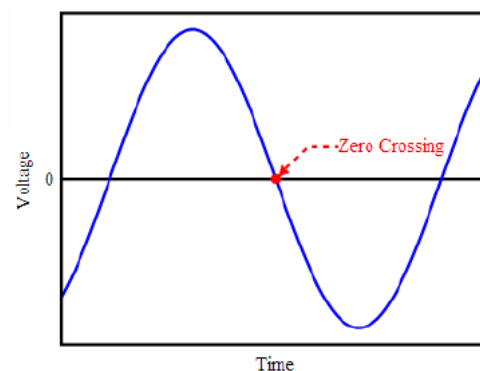
Transitions between consonants depend on how the consonants are interconnected.

Discrimination features:

- **VOT (Voice Time Onset)**: discriminates plosive consonants (consonant sounds that are formed by completely stopping airflow) like [p], [t], [k], [b], [d], [g]



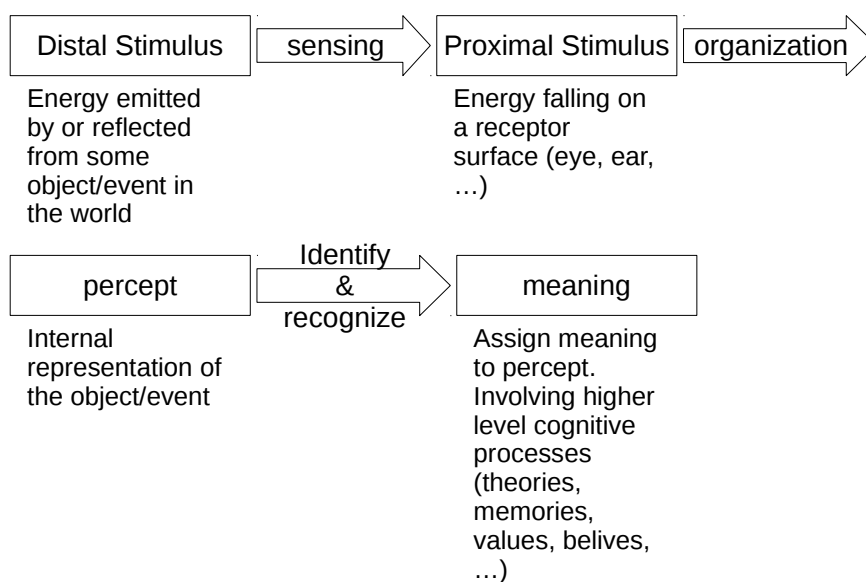
- **MFCC (Mel Frequency Cepstral Coefficients)**: that models the response of the human auditory system
- **Flatness**: indicates if sound is more noisy or musical (white noise = 1, pure tone = 0)
- **Harmonicity**: relationship between the noisy and the total part of a signal (through central spectroid)
- **Zero Crossing Gate**: number of zero crossings of the signal (useful to distinguish between voiced and unvoiced consonants)



Perception: prosody and voice

Perception process

Through the senses receive external input and then interpret them.



Focus and attention

Focus: determines what stimuli will be processed

World experience: through nature and learning

Selective attention:

1. focused stimuli becomes “highlighted” in memory
2. internal representations of unattended stimuli are suppressed

Preattentive processing: operates on sensory inputs before you attend to them, as they come into the brain from sensory receptors.

Language perception

1. **Audio vibrations** start nerve impulses
2. **Basilar membrane** (memory effect): filter at various frequencies. It adapts at what you are listening. Example: hearing bass sound, then need time to adapt to high frequency sounds.
3. **Cortis organ:** ciliated cells and nerve fibers transmit the sound potential to the brain (electricity). That it then processes the information.
4. **Brain :** acoustic nerve moves information from the cortis organ to the brain

Left hemisphere: elaborates structure and meaning of the language

1. *Broca area:* interpret nerve impulses
2. *Wernicke’s area:* understands meaning

Right hemisphere: perceives musicality

important in tonal languages (example: chinese), where intonation is phonologically relevant.

Cognitive theories

- **Top-down speech processing:** knowledge or expectations are used to guide processing
- **Bottom-up processing:** acoustic signals → words

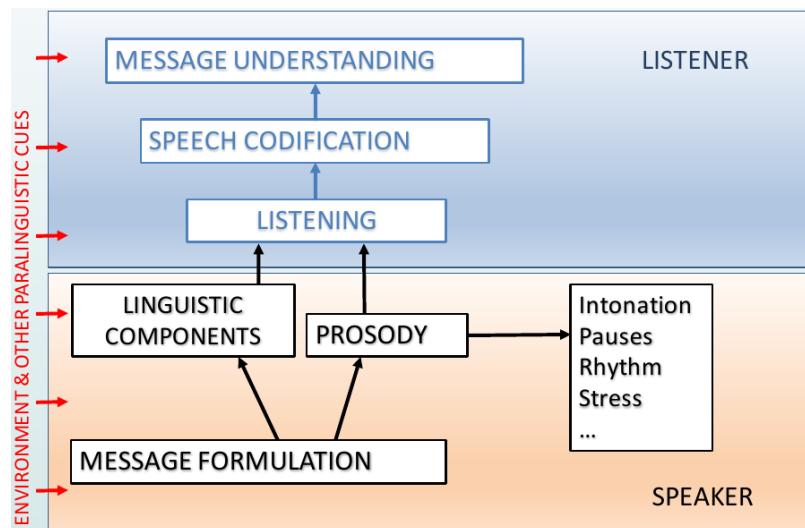
Prosody: prosody and voice descriptors in digital audio

Prosody: study of the tune and rhythm of speech and how these features contribute to meaning. Concerned with those elements of speech that are not individual phonetic segments (vowels and consonants) but are properties of syllables and larger units of speech.

At the phonetic level, prosody is characterized by:

- *vocal pitch* (intonation)
- *loudness* (acoustic intensity)
- *rhythm and rhythm* (phoneme and syllable duration)

Pragmatic: distinction between literal meaning and the intended meaning by the speaker of a sentence.



Intonation

Linguistically controlled and pragmatically meaningful use of F0 that spans entire expressions.

Autosegmental Metrical Theory (AM)

Connects stress and phrasing with intonation.

- **Tones (intonation)** → has high (H) or low (L) tones (primitives of intonation) concatenated in various configurations and associated with metrical structure (not a transcription system for intonation). Each tone adds a small element of meaning to the discourse as a whole.
 - represented as a string of auto-segments (example: LHLHLH)
 - independent of vowels and consonants
 - independent of each other
- **Autosegments (autosegmental)** → autosegments are elements that compose tones
 - *Pitch accents* (symbol: *)
 - H* - new information added to discourse
 - L+H* - information contrasting with prior item in the discourse
 - L* - not new information to be added to discourse, because implicit or extra-propositional
 - L+H* - question relevance of item to discourse in contrast to another item
 - *Phrase accents* (symbol: -), for intermediate phrases (one in each intonational phrase)
 - *Pitch boundaries* (symbol: %), for intonational phrases: continuous pitch contour delimited by boundary tones

example:

édìmbá bìsónjé ékúé òbô

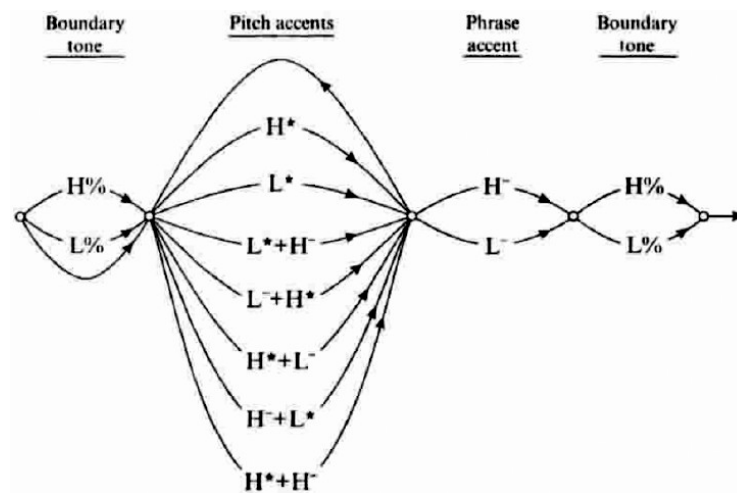
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

H L H L H H L HL

Tones and Break Indices (ToBI)

Family of systems designed for the prosodic annotation of spoken corpora (not a transcription

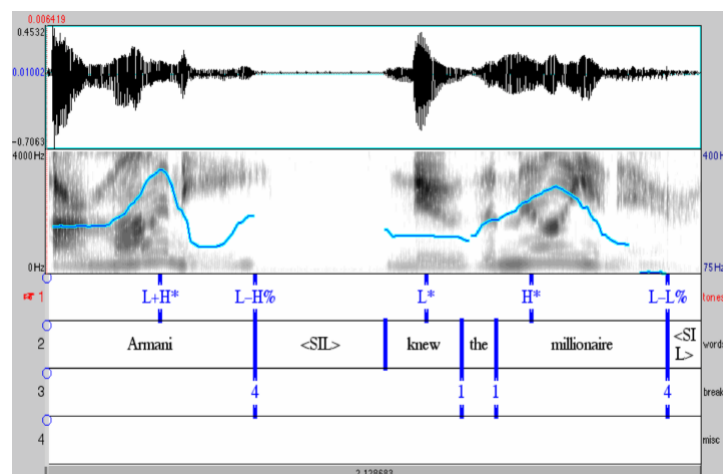
system for intonation). ToBI is language dependent.

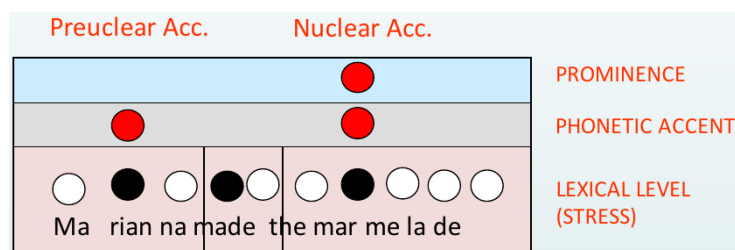


Goal: create searchable databases and extract regularities from the data

ToBI notation:

- acoustic waveform
- F0 contour
- labels tier
 - tone tier: H (high) and L (low) pitch targets for accentuation and phrasing
 - orthographic tier: straightforward transcription of all of the words
 - break index tier: 0-4; break indices represent a rating for the perceived degree of juncture between each pair of words and between the final word and the silence at the end of the utterance
 - a miscellaneous tier: * marks accentsPitch in intonation





Prominence (Nuclear accent): unique and most important accent of the phrase

Phonetic accent: language/dialect characterization, placed on stressed syllables

Stress: at syllable level

Coherence

Coherence: cohesion/connection among ideas in a discourse. Relative property.

- **Intonation** important in coherence (see how each autosegment can help)
- **gestures**
- building conversation by trying to understand **mental model of the other person** → intonation, syntax, semantics, gestures, ...

People in a discourse create **mental representation** of what has been already said, 3 important sources for mental representation:

- **Linguistic input:** anaphora, conjunctions, ...
- **Previous knowledge:** prosody, associating intonational morphemes with the lexicogrammatical structure of the text (new info? Already accessible info? ...)
- **Perception of physical environment** (gestures, deictic pronouns, ...)

Paratone

Prosodic equivalent of a written paragraph: speakers manipulate pitch, volume, tempo and pause at transition points between topical constituents to indicate relationship among topics.

- **High:** opening of a new organizational unit of the discourse
- **Low:** closing of an organizational unit of the discourse or marks an “aside”

Tempo and volume

Tempo and volume of a sentence can change meaning and characterize a speaker’s voice.

Stress and Rhythm

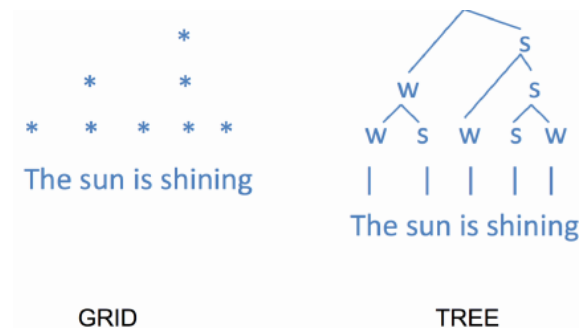
While rhythm gives an underlying organization to the language and the discourse, on which stress is built upon.

Stress

Relative prominence between perceived expressions thanks to a combination of phonetic parameters. Language dependent.

Stress is a phonological characteristic of lexical item and thus predictable.

Syllables designed as strong (s) or weak (w) and grouped in trees:



Rhythm

Independent between intonational phrases and speakers (instead it's language dependent).
Harmonizes the speech to be more approachable by a listener.

New phonological approach: people don't discriminate by rhythm but by speaking rate that changes in different contexts (example: doing a lecture).

Paralanguage

Meta communication that conveys meaning and emotions to what is being said:

- gestures, mimicry, volume, pitch variations

Used for: quoted speech and by storytellers

Phrasing

The way we choose to pronounce syntactic elements using paralinguistic cues (stops, pauses, melody envelopes, ...).

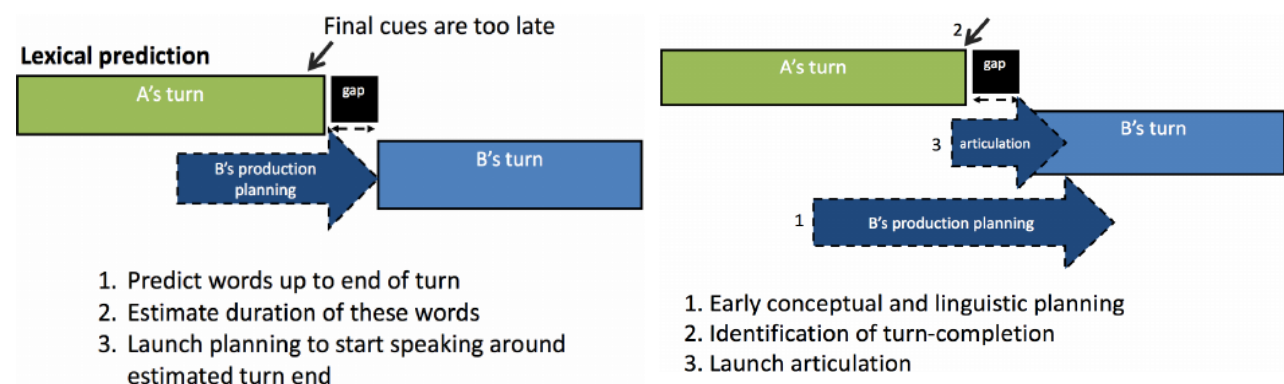
Sandhi: phonological process at morpheme or word boundaries that alters and fuses sounds

Speech Perception

Process by which the sounds of language are heard, interpreted and understood.

Prosodic features are produced and interpreted according to social, environmental and subjective rules.

Turn taking differs if response planning is taken early:



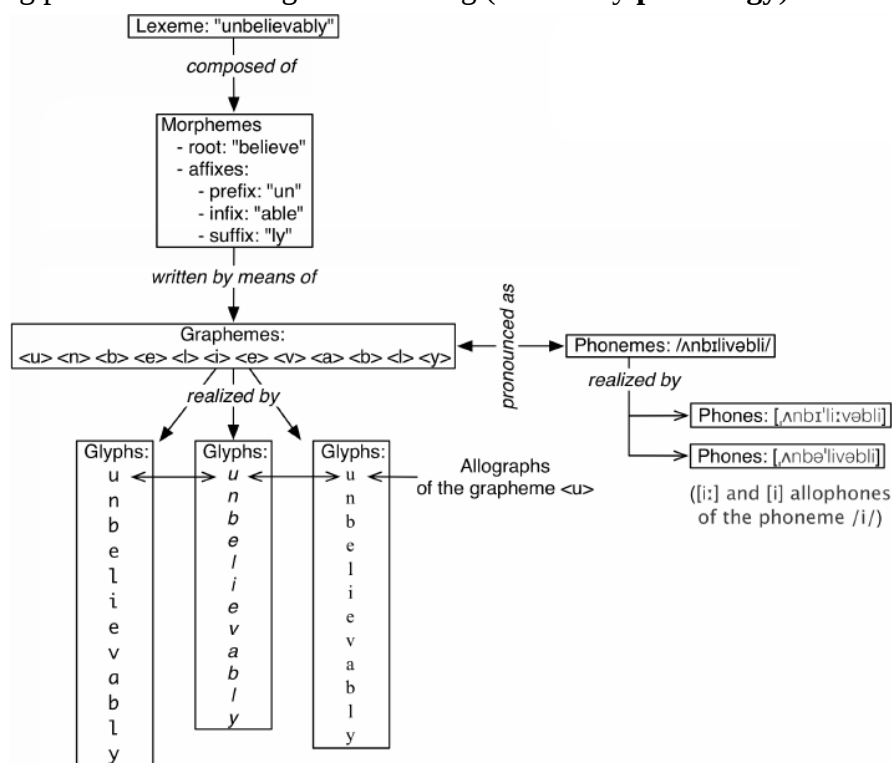
Speech processing

Intro

Phonetics and phonology

Words: pronounced and recognized as a series of symbols → phones. Language models are used to predict the concatenation of phones.

- **Phones:** small units that compose words (studied by **phonetics**)
composed by consonants and vowels.
- **Phonemes:** abstraction of a set of phones (allophones) which are perceived as equivalent. Switching phonemes do change the meaning (studied by **phonology**)



Human voice in the frequency domain

Voice: aperiodic signal which is continuous in the spectrum

Pitch and formants:

- **Pitch (F_0):** highest spike in the spectrum denotes the note (height of the sound)
- **Formants ($F_1, ..$):** spikes in the spectrum denotes the timbre (distinguishes person's voice) characterized by central frequency and bandwidth

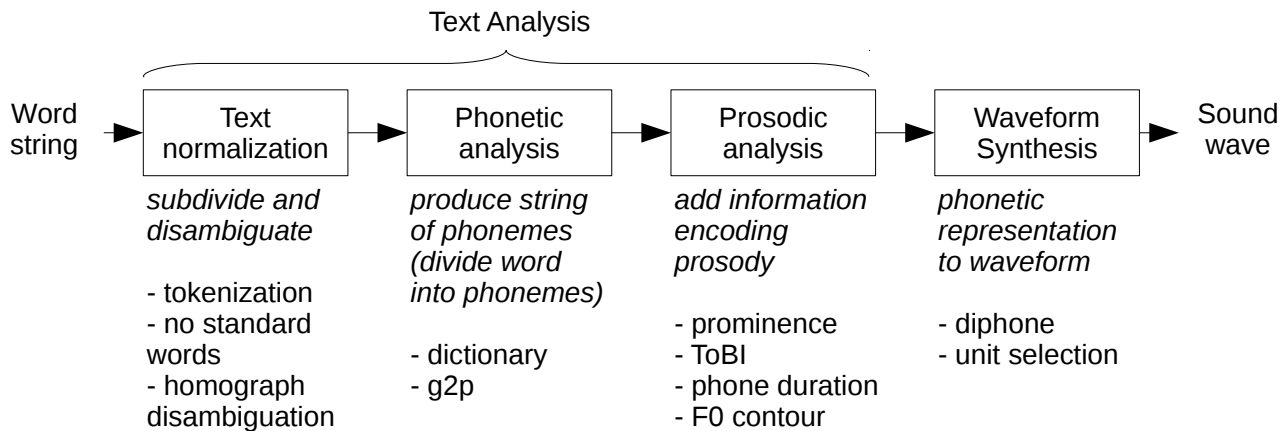
Vowels and consonants:

- **Vowels:** produced by the vibration of the vocal cords. Periodic signal. Distinguished by the first 3 formants.
- **Consonants:**

- voiced (/m/, /b/, ...): vocal chords vibration + noise → may have pitch + formants
- unvoiced (/t/, /c/, ...): noise → no pitch and formants

Intensity: measure loudness (db) → signal power

Text-To-Speech (TTS)



Goal: text string → waveform

steps:

1. **TEXT ANALYSIS:** text string → phonetic representation

1. **Text normalization:**

- *tokenization*: difficult without punctuation → Machine learning
- *no standard words*: numbers, abbreviations, ...
- *homograph disambiguation*: words written in the same way but with different pronunciation and meaning

2. **Phonetic analysis:** produce string of phonemes (divide word → phonemes).

Various ways to perform phonetic analysis:

- *dictionary*: contains phonetic representation of words
- *g2p*: for words not in dictionary (classifier)

3. **Prosodic analysis:** add information encoding prosody

adds intonation and rhythm to intermediate and intonation phrases, example:

[<I wanted> <to go> <to London> ,] [<but> <could only get tickets> <for France>]

intermediate
intonation

- *Prominence*: words more prominent in a phrase (via pitch, rhythm, energy).
Most prominent pitch → nuclear accent.
- *ToBI*: phonological theory of intonation, models prominence, tune, boundaries.

Pitch Accents		Boundary Tones	
H*	peak accent	L-L %	“final fall”: “declarative contour” of American English
L*	low accent	L-H %	continuation rise
L*+H	scooped accent	H-H %	“question rise”: cantonal yes-no question contour
L+H*	rising peak accent	H-L %	final level plateau (plateau because H- causes “upstep” of following)
H+!H*	step down		

- *Phone duration*: can be obtained through rules or ML
- *F0 contour*: define F0 for accents/breaks and then the contour interpolating found F0 values

2. WAVEFORM SYNTHESIS: phonetic representation → waveform

various ways to synthesize voice:

- formant synthesis: rules/filters to create voice through acoustic model + additive synthesis
- articulatory synthesis: model and simulate articulations and acoustics of vocal tract
- **concatenative synthesis**: most used. Concatenate small prerecorded wave units.

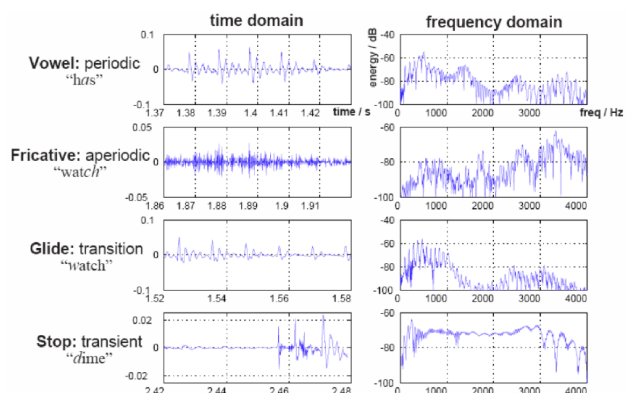
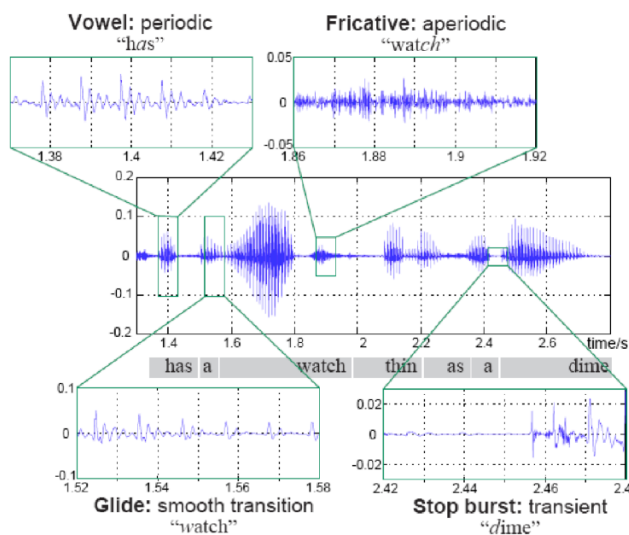
Two models:

- *Diphone*: phone-like that comprehend last part of a phone and first part of the successive phone → coarticulation: each phone differs slightly
- *Unit selection synthesis*: combine units (any piece of speech: diphones, syllables, ...) in a db using a cost function to determine best combination.

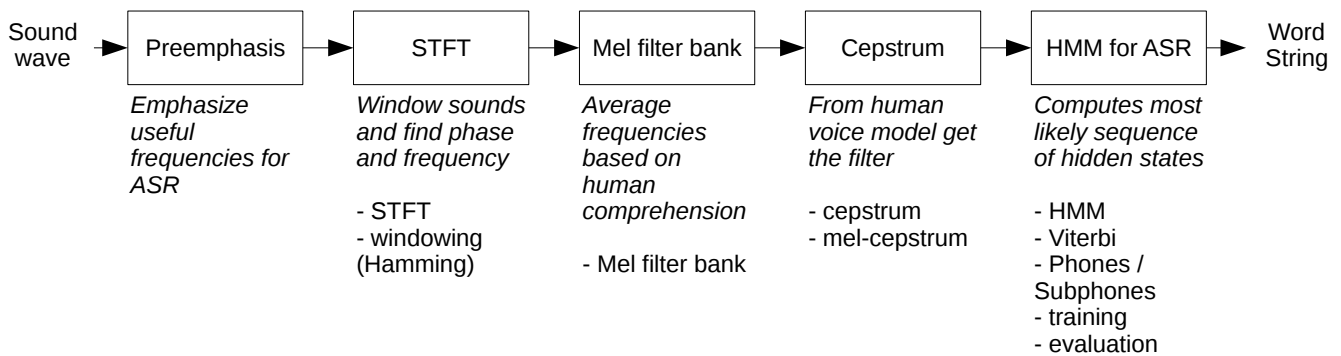
Automatic Speech Recognition (ASR)

Goal: speech to text

works on frequency and time domain:



Steps



1. Short time Fourier transform

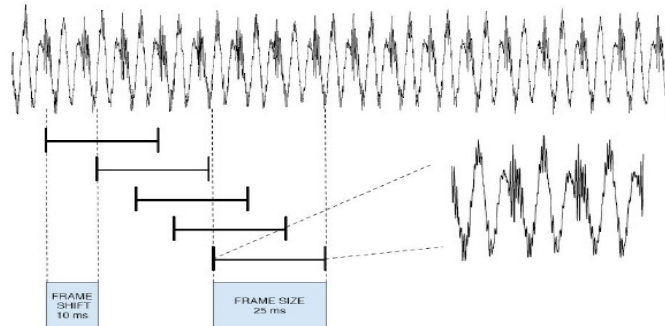
Vocal signal changes over time (formants are only valid for a brief time period) →

STFT: determines frequency + phase of local sections of a signal as it changes over time.

$$STFT_m[k]\{x_m\} = \sum_{n=0}^{L-1} x_m[n] \cdot w[n] \cdot e^{-j\frac{2\pi}{L}kn}$$

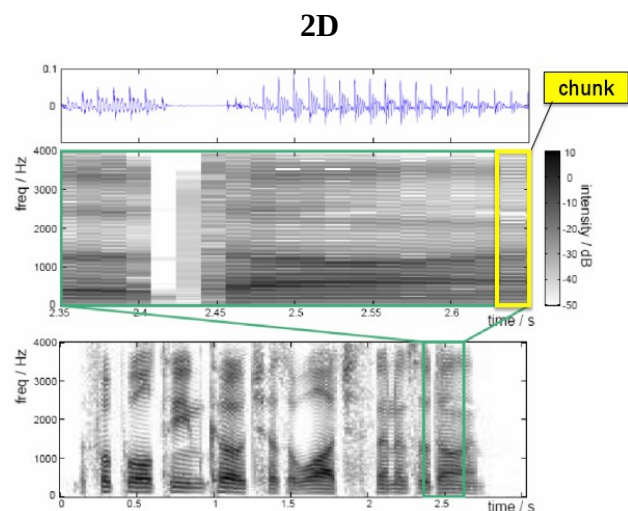
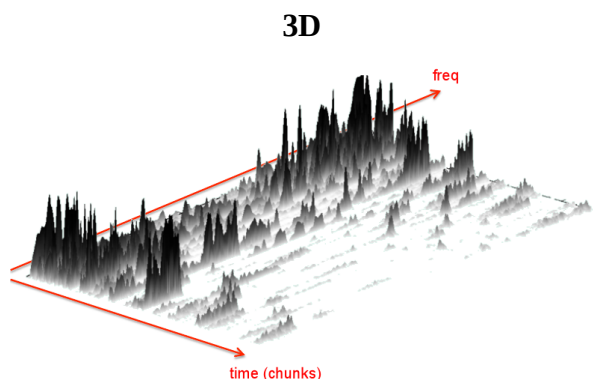
$STFT_m[k]\{x_m\}$ is an L-sized vector containing the Fourier transform of the chunk m (each k component is a complex number).

- $x_m[n]$ chunks (M # of chunks) of L samples
chunks are overlapped to further reduce the border effect



- $w[n]$ window function which reduces border effects of the chunks (most used is Hamming)

Spectrogram

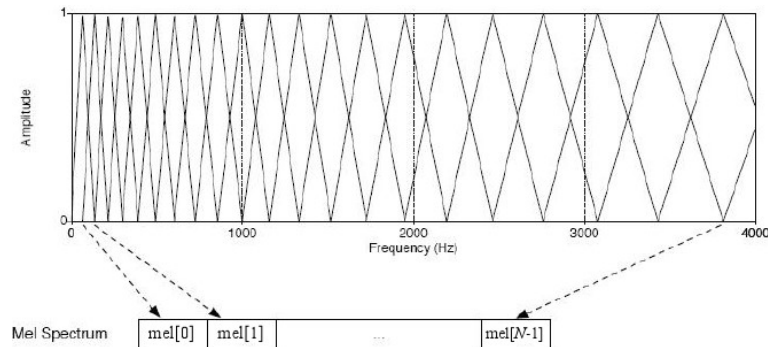


Color denotes the amplitude (the darker the higher)

2. Mel filter bank

Human do respond worse to higher frequencies → triangular filters which varies wrt to the frequency (bigger in higher frequencies). It averages higher frequencies more.

$$mel_m = mel_{filters}(STFT_m)$$



3. Cepstrum

Given the human voice generation model (source+filter), **only the filter do carries important info.**

Cepstrum separates filter component from the source:

1. takes waveform $x[n]$
2. compute STFT $X[k]$
3. compute STFT of the log of $X[k]$ (considered as a new waveform)
4. cepstrum coefficients are the amplitude of such new spectrum

Mel-cepstrum

Cepstrum applied to Mel filter bank.

MFCC coefficients (39 for each chunk):

- First mel-cepstrum coefficients [13]
 - Variations: deltas [13] and double deltas [13]
- } from which can be extracted the energy

4. Hidden Markov Model for ASR

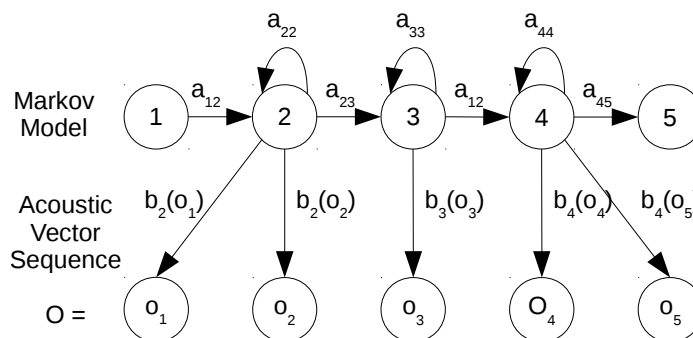
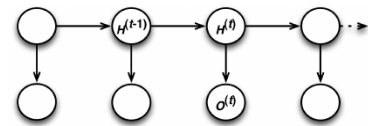
- $Q = \{s_1, \dots, s_{N_s}\}$ set of all subphones taken into consideration
- O set of observations (MFCC vectors)
- $A = \{a_{ij}\}_{(N_s \times N_s)}$ transition probability matrix (prob to move from subphone i to j)
- ~~intra-phone probabilities equal for all words~~
 - (not present in A since all are equal)
 - **inter-phone probabilities not equal for all words**
- $B = \{b_j(o_t)\}_{(N_s \times V)}$ emission probability matrix (prob that subphone j emits the observed MFCC vector o_t)

so $N_s = \underbrace{|Q|}_{\text{\# subphones}}$

$$\hat{h}^{1:n} = \underset{H^{1:n}}{\operatorname{argmax}} p(H^{1:n}, O^{1:n}) = \underset{H^{1:n}}{\operatorname{argmax}} \prod_t p(O^t, H^t) \cdot p(H^t, H^{t-1})$$

with:

$$p(H^{1:n}, O^{1:n}) = \frac{\left(\prod_t p(O^t | H^t) \right) \cdot \left(\prod_t p(H^t | H^{t-1}) \right)}{p(O^{1:n})}$$



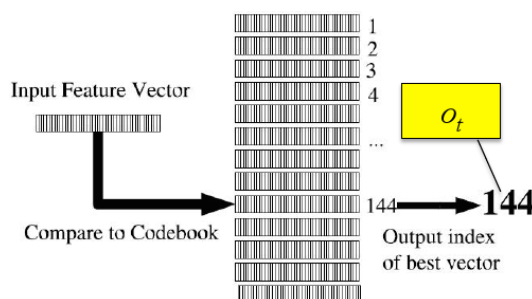
Viterbi algorithm computes $\hat{h}^{1:n}$ without exploring whole state space of $H^{1:n}$.

HMM problems:

- given model parameters, how to compute the most likely sequence of hidden states which could have generated a given observed sequence? → Viterbi algorithm (example: POS tagging)
- given model parameters, how to compute probability of a particular observed sequence? → Forward and Backward algorithms (Example: phoneme recognition)
- given output sequence, find most likely set of state transition and output probabilities → solved by Baum-Welch algorithm
- **independence and time-invariance assumptions** → **discriminative models**

2 approaches:

- discretize MFCC → **discrete HMM** (B is a matrix)



Codebook: list of all possible MFCC vectors $V = \{v_1, \dots, v_n\}$. Created through clustering a set of sample MFCC into V clusters.

Compute distance to compare codevectors in the codebook with a new MFCC to select closest.

Index of selected MFCC is the o_t

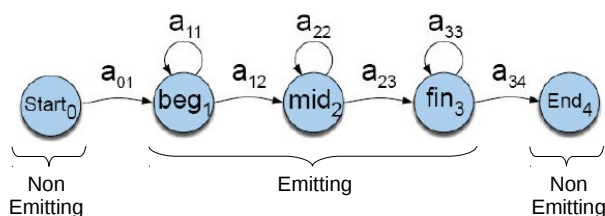
- continue MFCC → **Gaussian Mixture Model (GMM)** (no B matrix, $b_j(o_t)$ is a function and we store its parameters)

HMM with Gaussian distribution to model emission probabilities.

Models each MFCC vector with a single, multivariate Gaussian with 39 dimensions.

[GMM formula]

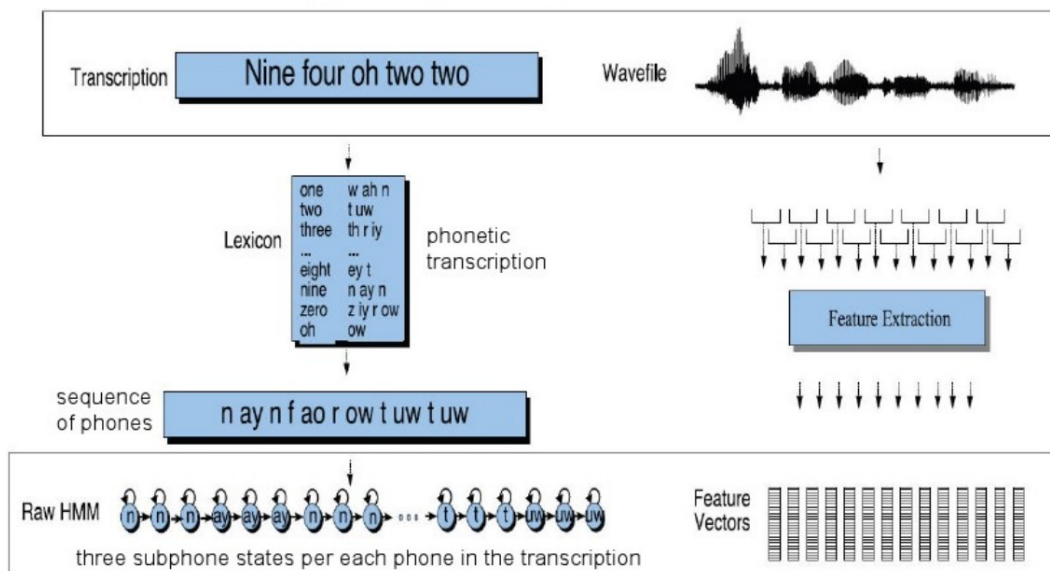
Phones and subphones



Each phone represented by 5 states.

a_{ij} transition probabilities

Example:



Training

Embedded training

Given: phoneset, pronunciation lexicon, transcribed wavefiles

1. Build **HMM for each sentence**
2. **Init A** probabilities to 0.5 (loopbacks) or 0 (all other transitions)
3. **Init B** probabilities by setting mean and variance for each Gaussian to:
 - GMM → the global mean and variance for the entire training set
 - HMM → random
4. run multiple iteration of Baum-Welch algorithm (expectation maximization algorithm, cannot use maximum likelihood because of hidden paths in HMM)

Decoding HMM/GMM

How to solve HMM and GMM

- **Viterbi algorithm:** dynamical programming algorithm that allows to compute the most probable path.

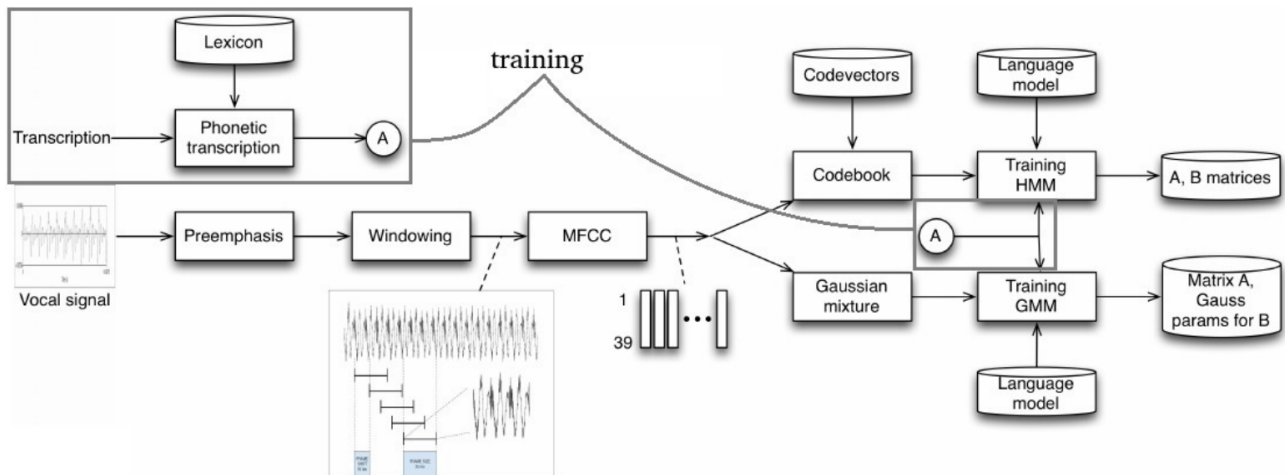
Not good for N-grams with $N > 2$

Alternatives: A* (ok for N-grams), Multiple pass decodings (break up decoding process in: collect best N-solutions and then apply more sophisticated algorithm on reduced space).

Problem: if ambiguous word with many exit arcs, Viterbi could favor an incorrect word.

- **Beam search:** parallel algorithm that during its execution cuts off less promising paths.

Recap



Evaluation

- **Word Error Rate (WER):** how much predicted word string differs from reference word
- **Sentence Error Rate (SER):** how many sentences had at least one error
- **Matched-Pair Sentence Segment Word Error (MAPSSWE):** compare 2 ARS's: align words generated by the 2 systems and then compute the P-value, if higher than a threshold → significant difference

Advanced ASR

- **Triphones:** phones varies based on phones on either side → triphone model
- **Posterior classifier:** integrate neural net or SVM to HMM architecture.
It computes $P(q_j|o_t)$ (q given o), but HMM needs reverse → Scaled likelihood (Bayes)
- **Noise:** additive noise → spectral subtraction, convolutional noise → cepstral mean normalization
- **Non words:** non verbal sounds (coughs, throat clearing, ...), environmental sounds (beeps, phones, ...) → for each of them create special phone and add to lexicon.
- **Speaker adaptation:** modify the acoustic model wrt a specific voice.

MLLR: start with generic voices acoustic model, and then learn a linear transformation matrix W and a bias vector (one for all phones or one for each phone), where:

$$\hat{\mu}^{(j)} = W \cdot \mu^{(j)} + \omega$$

Pragmatics

Dialogue and conversational agents

Linguistics of Conversation

Eliza: simple simulation of dialogues based on psychotherapeutic sessions, based on simple substitutions using regex: "s/<read text>/<text to substitute>".

Example:

“He says I’m depressed much of the time”



- Substitute I’m with YOU ARE
- s/. * YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/

“ I AM SORRY TO HEAR YOU ARE DEPRESSED.”

Basic Conversational Agents

Groups all the systems that can sustain a conversation in natural language.

Turn-taking

Dialogue characterized by turn taking: A: ... → B: ... → A: ... → [...]

Rules based on turns to decide who and when someone can speak:

- If current speaker decides B should speak → B can take the next turn
selection of next speaker based on utterances (adjacency pairs: question/answer, greeting/greeting, ...)
- If current speaker doesn’t decide who should speak → anyone can take next turn
- If no one takes next turn → current speaker may take next turn

Speech Acts

Speaker performs 3 simultaneous acts in issuing an utterance:

- **locutionary act:** act of saying something with a certain sense and reference
example: question, declarative, imperative, ...
example: “He said to me ‘Shoot her!’ meaning by ‘shoot’ shoot and referring by ‘her’ to her”
- **illocutionary act:** act performed in saying something (act named and identified by the performative verb)
categories:
 - Assertives: committing the speaker to something’s being the case
 - Directives: attempts by the speaker to get the addressee to do something
 - Commissives: committing the speaker to some future course of action
 - Expressives: expressing the psychological state of the speaker about a state of affairs
 - Declarations: bringing about a different state of the world via the utteranceexample: “He urged (or advised, ordered, & c.) me to shoot her ”
- **perlocutionary act:** act performed, or as a consequence of, saying something
example: “He persuaded me to shoot her”

Grounding

Dialogue is a collective act performed by speaker and hearer.

Common ground: set of things mutually believed by both speaker and hearer.

Principle of closure: agent performing an action needs feedback on outcome. How to achieve common ground?

- **Continued attention:** B continues attending A
- **Relevant next contribution:** B starts in on next relevant contribution
- **Acknowledgement:** B nods or says continuer like “uh-huh”
- **Demonstration:** B demonstrates understanding A by paraphrasing or reformulating A’s contribution or by completing A’s utterances
- **Display:** B displays verbatim all or part of A’s presentation

Conversational Structure

Each medium has its own structure:

1. Enter a conversation
2. Identification
3. Establish joint willingness to converse
4. First topic is raised, usually by caller

Implicature

In a response there may be implicit information which need to be inferred by the other party.

Example:

A: “And, what day in May did you want to travel?”

B: “OK, uh, I need to be there for a meeting that’s from the 12th to the 15th.”



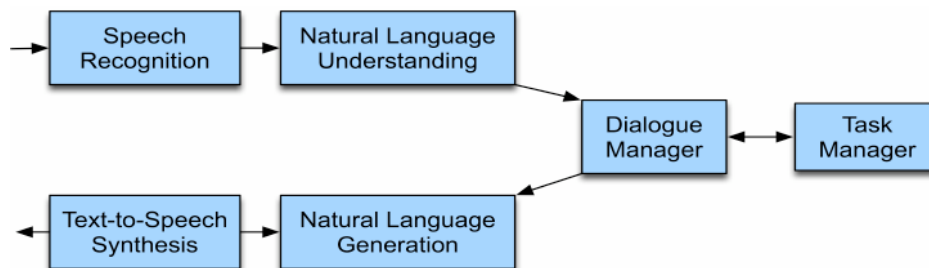
Cooperative principle:

inference is possible because of the underlying cooperativeness between the actors of the conversation.

Properties to respect to have relevant implications:

- **Relevance:** be relevant
- **Quantity:** do not under/over inform the other party
- **Quality:** do not mislead with false information
- **Manner:** avoid ambiguity and obscurity

Dialogue System Architecture



Speech recognition (ASR)

Speech to text process composed by:

- **recognizer for phones**
- **pronunciation** dictionary (how to pronounce a word)
- **grammar** that models the language and suggest which words will likely follow the current one
- **search algorithm** to find best string of words

Natural Language Understanding (NLU)

Many ways to represent sentence's meaning (computational semantics).

Frame and slots semantics is the most used representation for the meaning of the sentences, which can be generated by semantic analysis (semantic grammars, syntax driven, information extraction).

Natural Language Generation

1. **Content Planner + (Dialogue Manager):** decides meaning to express ("what to say")
2. **Language Generator:** chooses words to express meaning and the prosody ("how to say it")
 - sentence planner
 - surface realizer
 - prosody assigner

Text to speech synthesis

1. Takes words and prosodic annotations
2. Synthesizes a waveform

Dialogue Manager

Controls the architecture and structure of the dialogue by maintaining a state. It interfaces with the task manager. Possible dialogue architectures are:

- Finite State
- Information State (MDP)
- Frame-based
- AI Planning

Finite State

System completely controls (initiative on the system) the conversation with the user. It asks questions and ignoring or misinterpreting not related content in the user responses.

Initiative: denotes who has control over the conversation, the system, the user or alternatively both (as human dialogues are).

- **System initiative:** simple to build (topic → NLU and words → ASR always known to the system) and predictable for the user, but limited applications.
- **User initiative:** user directs the system, user ask question and system responds (example: web search query, db query).
- **Single initiative + universalis:** more flexibility with universalis (commands that can be used anywhere).
- **Mixed initiative:** conversational initiative shifted between system and user.

Example: simple implementation is by using the structure of the frame itself to guide the dialogue.

- User can answer multiple questions, but the system must understand which questions are answered and not to propose them again to the user.
- Avoid strict order of FSA architecture

Initiative type can be expressed through:

	Open prompt user answer are lightly constrained	Directive prompt user answers are heavily constrained
Restrictive language model strongly constrains ASR based on dialogue state	NA	System initiative
Non-restrictive language model not restricted by dialogue state	User initiative	Mixed initiative

Frame-based

Each discussion object which has it's own legs must be treated separately as discussion themselves.

Need to switch from frame to frame based on what user says and to disambiguate which slot of which frame an input is supposed to fill.

Main implementation through rules.

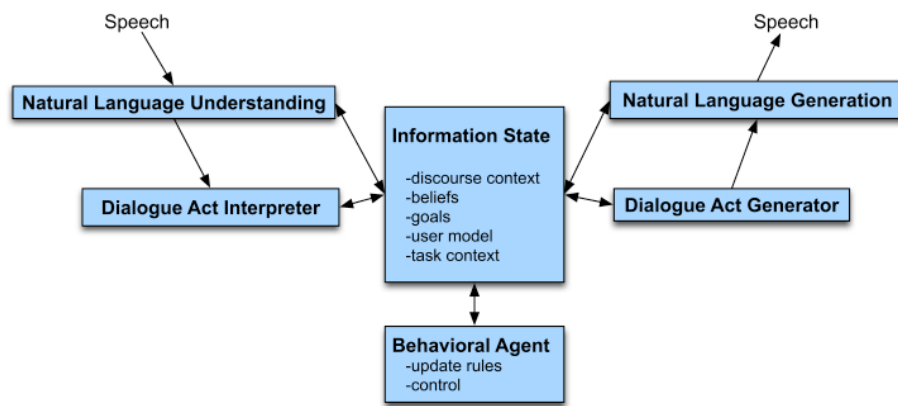
VoiceXML

XML-based dialogue design language. Deals with simple, frame based mixed initiative dialogue.

Information state

Information state architecture: more complex system of generation/interpretation than form-filling needs to:

- understand if user has made: question, proposal, rejected a suggestion
- ground user's utterance, ask clarification questions, suggestion plans



Composed by:

Dialogue-act interpreter

Act with (internal) structure related specifically to its dialogue function and grounding.

Dialogue functions can be grouped and conceived in various ways:

- **Verbmobil task:** thank, greet, introduce, bye, request-comment, suggest, reject, accept, ...
- **DAMSL:** divides in forward (statement, info-request, check) and backward (agreement, answer, understanding) looking functions
- **Traum & Hikelman:** turn-taking, grounding, core speech acts, argumentation

Given utterance how to assign correct dialogue act (Illocutionary force)?

- **Syntactic form:** yes-no questions → have auxiliary before subject syntax, statements → have declarative syntax, command → have imperative syntax.

But ambiguities (example: requests that looks like questions), the surface form is different from the speech act type:

	Locutionary force	Illocutionary force
<i>Can I have the rest of your sandwich?</i>	Question	Request
<i>I want the rest of your sandwich</i>	Declarative	Request
<i>Give me your sandwich!</i>	Imperative	Request

- **Statistical classification:** classification task with N classes (N dialogue acts) with 3 probabilistic models (3 cue types):
 - *words and collocations:* “please”/“would you” → request, “are you” → info-request
 - *prosody:* rising pitch → info-request
 - *conversational structure:* “Yeah” following proposal → agreement

Correction

Correction needed if system not sure about an interpretation, system can:

- **reject:** reject interpretation
 - **display misunderstanding:** and ask for a confirmation
- } user can: repeat, rephrase, say
no to confirmation

corrections are understood through ML: lexical information (words “no”, “correction”, “I don’t”, swear words), prosodic features, ASR confidence, length, repetition, ...

Dialogue-act generator

Confirmation

Need more grounding and confirmations than humans because ASR are worse than human.

- **Implicit confirmation:** display best if system guesses right, it's quicker and subtle
- **Explicit confirmation:** rephrase a summary of the choice and wait for reply

modern systems are adaptive, can switch between implicit and explicit confirmation based on confidence level.

Rejection

When confidence level is low.

Evaluation

How to evaluate algorithms? 2 ways:

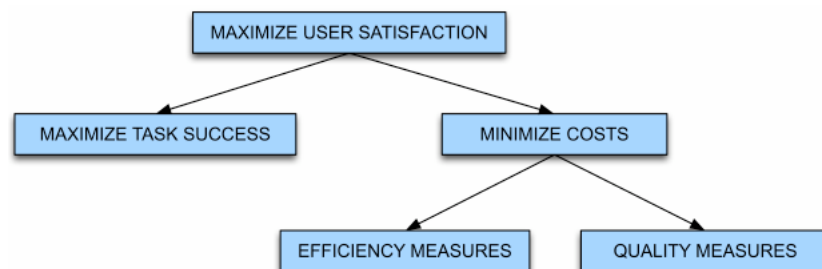
- **Extrinsic:** embedded in some external task
- **Intrinsic:** some sort of more local evaluation

Why evaluate?

- Metric to help compare different implementations
- Metric for how good a dialogue went → automatic improvement via reinforcement learning

Paradise evaluation

Maximize task success and minimize costs (efficiency and quality measures).



Maximize task success: considered as

- % of subtasks completed
- correctness of each questions/answer/error msg
- correctness of total solution
- user perception

Minimize costs:

- Efficiency measures:
 - total elapsed time-invariance
 - number of queries
 - turn correction ratio
- Quality measures:
 - number of times ASR failed to return any sentence
 - number of ASR rejection prompts
 - number of times user had to barge in
 - inappropriateness

After dialogue questionnaire for user satisfaction rating

Plan based dialogue agents

Utility based

Policy strategy

Metric used to drive the learning: learn optimal policy for how the conversational agent should behave.

Conversational agent characterized by:

- **S**: set of states that represent the current knowledge of the system
- **A**: set of actions that the agent can perform
- **G**: goal which is the “user satisfaction” which implies:
 - *success metric*: how well agent performed to achieve goal
 - *policy* π : use success metric to create a policy that describes what action to perform being in a particular state (strategy).examples: when to ground/confirm/reject/ask for clarification, ask directive prompt, use user/system/mixed initiative.
other strategy considerations do involve when to use user/system/mixed initiative (based on: open vs directive prompts, restrictive vs non-restrictive grammars).

- **Utility**: maps state/state sequence \rightarrow real number (represents how good a state is)

rational agent should choose an action that maximizes the agent Expected Utility.

$$EU(A|E) = \sum_i P(Result_i(A)|Do(A), E) \cdot U(Result_i(A))$$

iterates over each state that can be reached by applying A

where:

- E : current state where the agent is
- A : action to perform
- $Result_i(A)$: state resulting in applying action A

Markov Decision Process (MDP)

Dialogue as trajectory in space ($S_1 \rightarrow a_1, r_1 S_2 \rightarrow a_2, r_2 S_3 \rightarrow \dots$) and best policy π^* is the one with greatest expected reward over all trajectories.

Based on the policy strategy already seen:

- **Reward** $r(a, s)$: that agent receives for taking an action in a state.

How to compute reward?

Discounted rewards $\gamma \in [0, 1]$: cumulative reward of a sequence as the discounted sum of utilities of the individual states $Q([s_0, a_0, s_1, a_1, \dots]) = \gamma^0 \cdot R(s_0, a_0) + \gamma^1 \cdot R(s_1, a_1) + \dots$. The higher γ the greedier. Which can be computed with:

Bellman equation

$$Q(s, a)(s) = \max_{a \in A} \left[\underbrace{r(s, a)}_{\text{immediate reward}} + \lambda \sum_{s' \in S} p(s'|s, a) \cdot \underbrace{\max_{a'} Q(s', a')}_{\text{future rewards}} \right]$$

- if labelled data: $p(s'|s, a) = \frac{C(s, s', a)}{C(s, a)}$ if we have labelled data
- if not labelled data: hand-tune small set of states and policies and generate simulated conversations or get real ones

value iteration algorithm performed to learn Q values → best policy

- **State:** each state could have all the information (ASR confidence, questions, answers, ...) of the dialogue so far, but usually limited set to recent information.
- **Actions:** speech acts such as: ask question, explicit confirmation, rejection, tell info, tell choices

markov transitions assumption: $P(s_{t+1}|s_t, s_{t-1}, \dots, s_0, a_t, a_{t-1}, \dots, a_0) = P_T(s_{t+1}|s_t, a_t)$ all that matters is the state immediately before not the previous ones.

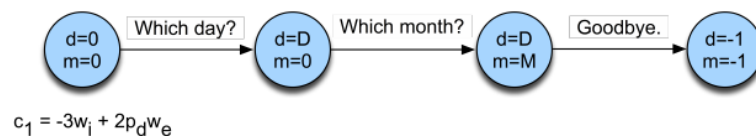
“Day and Month dialogue” system example

Goal: fill 2 slot frame: Month, Day with minimum interaction with the user.

- **States:**
 - special states s_i (initial state), s_f (final state)
 - 365 states with day+month
 - 1 state for leap year
 - 12 states with month
 - 31 states with day
- **Actions:**
 - a_d question asking for the day
 - a_{dm} question asking for the day+month
 - a_f submit form and close dialog
- **Reward:** cost function for the entire dialogue

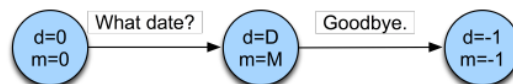
$$C = w_i \cdot \underbrace{N_i}_{\text{\# of interactions (diag duration)}} + w_e \cdot \underbrace{N_e}_{\text{\# of errors in obtained values (0-2)}} + w_f \cdot \underbrace{N_f}_{\text{expected distance from goal (0-2)}}$$
- **Policies:** policy 1 better when improved error rate justifies longer interaction

Policy 1 (directive)



$$c_1 = -3w_i + 2p_d w_e$$

Policy 2 (open)



$$c_2 = -2w_i + 2p_o w_e$$

with: P_d probability of error in directive prompt, P_o probability of error in open prompt

Normally search space for policy very large → reinforcement learning

Planning

Agent uses **inference rules** to understand **what caused user to say what he said** (after hearing).

Plan based models of dialogue are referred as BDI models since they need to model:

- **Beliefs:** what the agent knows $KNOW(\underbrace{S}_{\text{agent}}, \underbrace{P}_{\text{what agent believes}})$
- **Desire:** relies on predicate $WANT(\underbrace{S}_{\text{agent}}, \underbrace{P}_{\text{what agent wants (state/exec of some action)}})$
- **Intentions**

Plan inferential interpretation and production

Action schemas based on STRIPS are used to axiomize actions and planning (required for BDI systems).

- **Preconditions:** conditions that must be true to perform the action
- **Effects:** conditions that become true after performing action
- **Body:** partially ordered goal states that must be achieved in performing the action

Example:

C1: I need to travel in May.

A1: And, what day in May did you want to travel?

C2: OK uh I need to be there for a meeting that's from the 12th to the 15th

Meeting → relevant to flight booking
precondition to have meeting → being at place where meeting is held
being at place → flying there

User wants to fly there before the 12th of May

Intentions:

1. **Intentional structure:** dialogue is coherent if each utterance has an underlying plan-based intention of the speaker.
2. **Discourse Purpose (DP):** intentions are instantiated in the model assuming that each discourse has an underlying purpose held by the person that initiates it.
3. **Discourse Segment Purpose (DSP):** each discourse segment has its corresponding purpose.

Only dominance and satisfaction-precedence coherence relations are used:

- DSP1 dominates DSP2 if part of DSP1 satisfaction comes from DSP2
- DSP1 satisfaction precedes DSP2 if DSP1 must be satisfied before DSP2

Example:

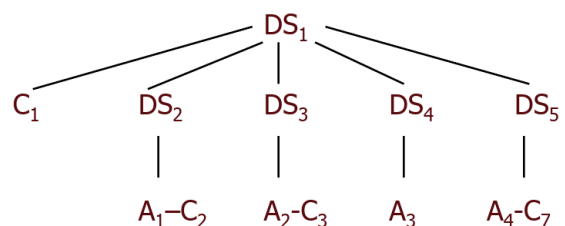
I1: (Intend C (Intend A (A find a flight for C)))

I2: (Intend A (Intend C (Tell C A departure date)))

I3: (Intend A (Intend C (Tell C A destination city)))

I4: (Intend A (Intend C (Tell C A departure time)))

I5: (Intend C (Intend A (A find a nonstop flight for C)))



Computing with Affective Lexicons

Lexical semantics can be modeled wrt:

- Emotion
- Mood
- Interpersonal stance
- Attitudes/Sentiment
- Personality

Why?

- **Detect:** sentiments towards entities, frustration of callers, stress in drivers, medical conditions, emotions in novels, ...
- **Generate:** emotion/moods in videogames and storybooks, personalities for dialogue systems

Words can have connotations other than sense.

Sherer affective states

Sharer has categorized affective states.

Name	Duration	Description	Example
<i>Emotion</i>	Short	Synced response to evaluation of a significant event	Anger, sad, joy, shame, fear, ...
<i>Mood</i>	Long	Change in subject feeling with no apparent cause	Cheerful, gloomy, irritable, depressed, ...
<i>Attitudes / Sentiment</i>	Mid	Affective and preference predisposition towards object/person	Liking, loving, hating, ...
<i>Personality traits</i>	Stable	Stable personality dispositions and behavior	Nervous, anxious, reckless, ...

Connotation of affectiveness in lexicon:

- connotation lexicon built by: hand, semi-supervised with seed words, fully supervised
- supervised (learn weight for each lexical category) and unsupervised (pick simple majority sentiment)

Attitudes/Sentiment

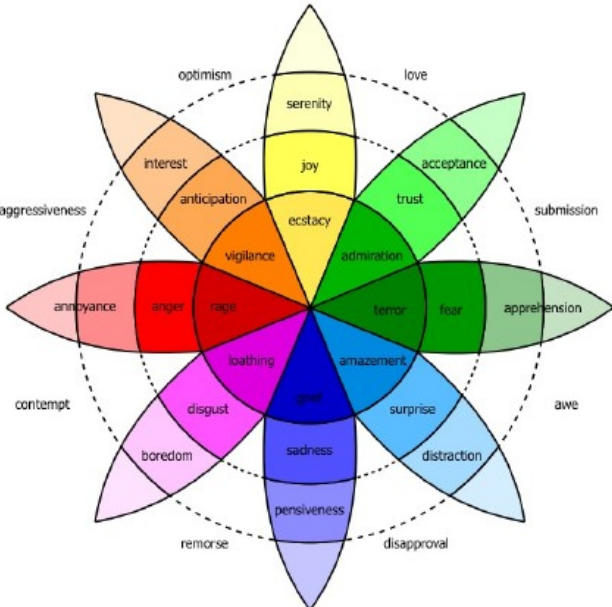
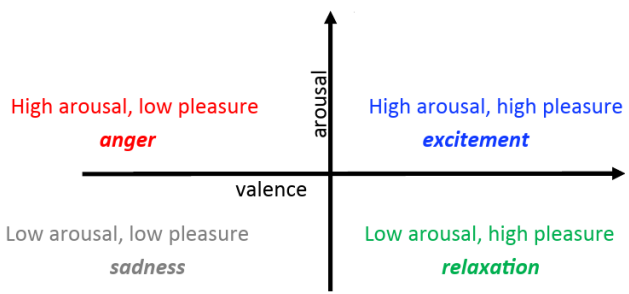
Implementations:

- **General Inquirer:**
 - divides vocabulary in positive and negative words
 - strong vs weak, active vs passive, overstated vs understated
 - pleasure, pain, virtue, vice, ...
- **LIWC (Linguistic Inquiry and Word Count):**
 - distinguishes between positive and negative emotions
 - cognitive processes
 - pronouns, negation quantifiers
- **MPQA Subjectivity Cues Lexicon:**
 - divides vocabulary in positive and negative words

- **Bing Liu Opinion Lexicon:**
 - divides vocabulary in positive and negative words
- **SentiWordNet**
 - WordNet synsets automatically annotated in positive and negative words in a fuzzy way

Emotion

2 families of theories on emotions:

Atomic	Dimensions
<p>Finite list of emotions from which other are generated.</p> <p>Example (Ekman basic emotions): surprise, happiness, anger, fear, disgust, sadness</p>	<p>Determined by a combination of different values.</p> <p>Example:</p> <ul style="list-style-type: none"> • Valence: pleasantness of the stimulus (positive, negative) • Arousal: intensity of the emotion provoked by the stimulus (strong, weak) • Dominance: degree of control exerted by the stimulus
Emotions are units	Emotions are dimensions
Limited number of basic emotions	Limited number of label but infinite number of emotions
Basic emotions are innate and universal	Emotions are culturally learned
	

Lexicons for detecting document affect

- **Unsupervised:**
 - *Sentiment:*

- sum the weights of each + word in the document
 - sum the weights of each - word in the document
- } choose greater absolute value
- *Emotion*:
 - do the same for each emotion lexicon
 - **Supervised**:
 - *build a classifiers*: predict sentiment given features (using counts of lexicons categories)
 - *baseline*: use counts of all the words and bigrams in the training set, good only if training set is similar to the test one

Personality traits

Big 5 dimensions of personality:

- **extraversion vs introversion**
- **emotional stability vs neuroticism**
- **agreeableness vs disagreeable**
- **conscientiousness vs unconscientious**
- **openness to experience**

classifier with:

- labelled corpus for personality of author and normalize the word counts

Interpersonal stance

Logistic regression classifier with:

- LIWC lexicons (+ others lexical features)
 - prosody
 - discourse features
- interruptions and dialogue acts/adjacency pairs

Chatbots and Conversational agents

Different types of chatbots

- **Retrieval based vs Generative models**

Retrieval based models (easy): predefined responses + heuristic (rule based vs ML)

Generative models (hard): new responses from scratch (Machine Translation, translate from input to output)

No grammatical mistakes

yes

No handling of unseen cases

Yes, they can generalize

No reference on previous said things

yes

No training

Hard to train

- **Long vs short conversations:**

Long (difficult)

Short (easy)

Difficult to automate

Multiple turns

- **Open domain vs closed domain:**

Open domain (difficult)

Closed domain (easy)

Can take conversation anywhere → infinite number of topics

Only on topic

- **Coherent personality:** produce consistent answers to semantically identical inputs. Problem when training on very different sources.
- **Evaluation of models:** whether fulfills task or not.
- **Intention and diversity:** generative systems produce generic responses.

Discourse

Anaphora Resolution

Reference

“John went to Bill’s car dealership to check out an Acura Integra. He looked at it for half an hour.”

Reference: process by which speakers use words “John” and “he” to denote a particular person.

- *Referring expression:* John, he
- *Referent (abstract concept to which are related the referring expressions):* the actual entity, but as a shorthand we might call “John” the referent
- *Corefer (when 2 or more expression refer to same thing):* John, he
- *Antecedent (expression referred from a later appearing word):* John
- *Anaphor (expression whose interpretation depends on another expression in context):* he

Various types of references:

- speech activities
- manner of description
- proposition
- ...

Reference Phenomena

References can be made in various ways:

- **definite and indefinite noun phrases:** definite noun phrases preclude asking "which one?"
- **pronouns:** compared to definite noun phrases, pronouns require more referent *salience* (cataphora: if it precedes the noun it refers to).

2 approaches are possible wrt to constraints to resolve the antecedent:

- *hard constraints on reference:* number, gender, person&case (we), syntactic (him, himself)
- *soft constraints on reference:* selectional restrictions (semantic restrictions that a word imposes on the environment in which it occurs), recency (refers to the nearest possible

antecedent), grammatical role (subject preference: refers to the subject of the previous sentence), repeated mention (?), parallelism preference, verb semantics preferences

algorithms which use these constraints:

Lappin and Leass: given he/she/it it returns an antecedent by using recency and syntactic preferences. 1. when new noun phrase encountered → add representation with a salience value 2. choose most salient antecedent.

- **inferables:** infer a noun from another different noun
- **generics:** refers to the referent instead of the antecedent

Coherence

Text is coherent if:

- **Rhetorical structure:** appropriate use of coherence relations between subparts of the discourse.
- **Discourse structure:** appropriate sequencing of subparts of the discourse.
- Appropriate use of referring expressions.

Coherence relations between 2 sentences s0 and s1:

- result: state/event asserted by a0 causes/could cause state/event asserted by s1
- explanation: state/event in s1 causes or could cause state/event in s0
- parallel: infer properties from s0 and s1 being s0 and s1 similar
- elaboration: s0 as s1 in a more specific way
- occasion: change of state in s0 inferred from s0 or vice-versa

Discourse Structure

Each sentence can be taken and relationships can be represented as a tree:

John went to the bank to deposit his paycheck.

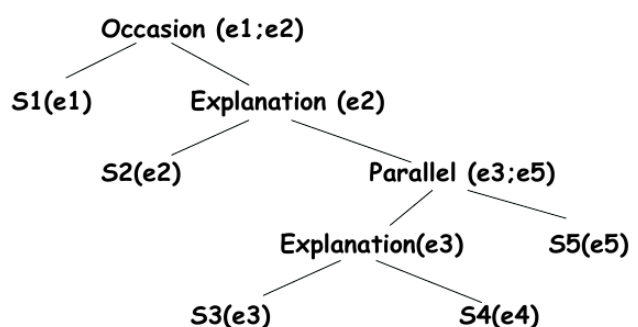
(S1)

He then took a train to Bill's car dealership. (S2)

He needed to buy a car. (S3)

The company he works for now isn't near any public transportation. (S4)

He also wanted to talk to bill about their softball league. (S5)



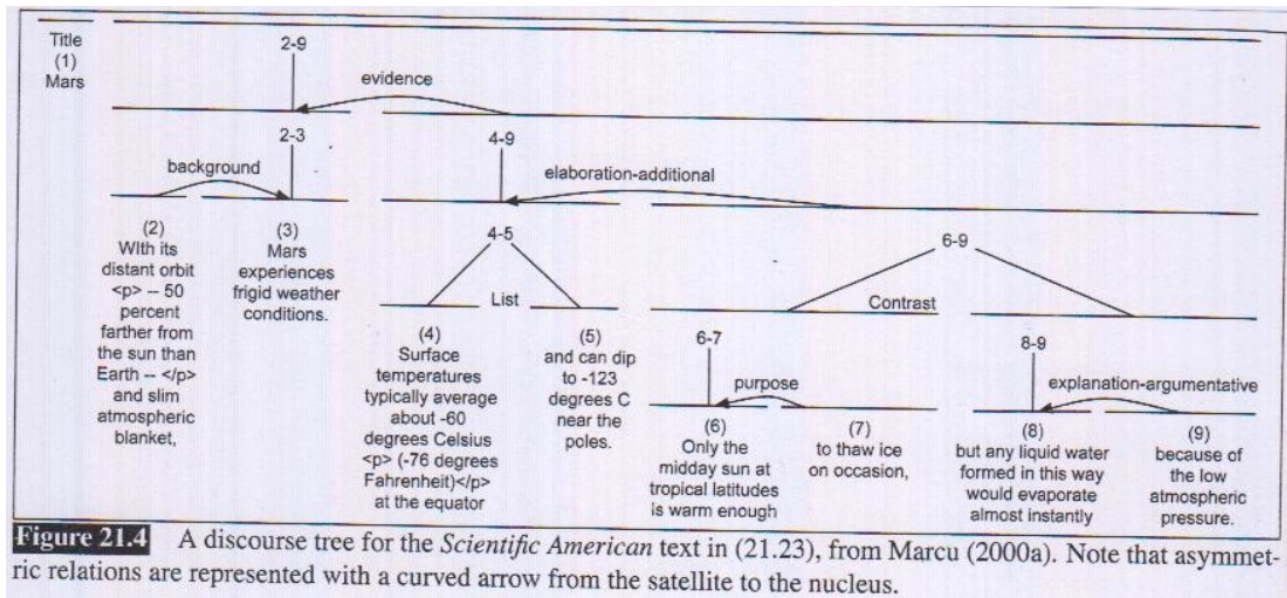
Rhetorical structure

Identifies relations between nucleus and it's attached satellites. Can also decide to only keep the nucleus.

Rhetorical relations between nucleus and attached satellites:

- elaboration: set/member, class/instance, whole/part
- contrast: multinuclear

- condition: satellite presents precondition for N
- purpose: satellite presents goal of the activity in N
- background: satellite gives context for interpreting N
- attribution: multinuclear
- list: multinuclear
- evidence: nuclear acquires truthfulness based on satellite



Automatic tagging can be performed with supervised ML. Features:

- explicit markers (because, however, therefore, ...)
- syntactic structures
- ordering
- tense/aspect
- intonation

Summarization and QA

Summarization

Process of distilling most important information from a text.

Various tipologies do exist: outlines, abstract, headlines, snippets, answers, summaries.

Classification

- **Single-document vs multiple-document summarization**
 single document: headline or outline
 multiple documents: condensation of the group
- **Generic summarization vs query-focused**
 generic: no particular user or info is needed
 query focused: need to extract particular information from the text

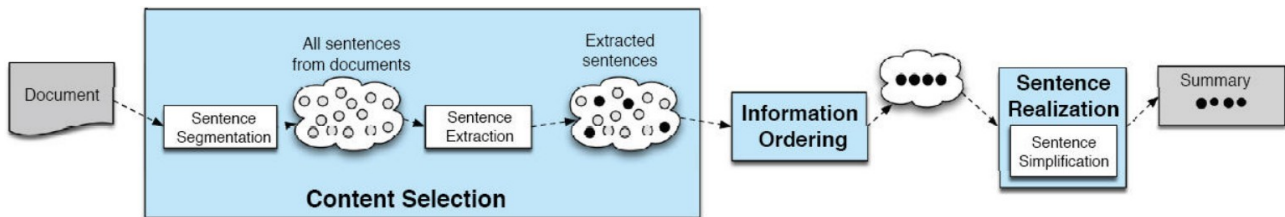
- **Abstract vs extract summarization**

extraction: select sentences or words from text

abstract: re-elaborate the text producing sentences (complex)

Single document

Summarize from a single document.



1. **Content Selection:** choose level of text granularity (sentence, paragraph, ...)
2. **Information Ordering:** choose the order of the extracted (usually keep appearance order)
3. **Sentence Realization:** clean extracted units to better fit them in the new context

Content Selection

To classify each sentence as important/unimportant classes can apply supervised or unsupervised content selection.

Unsupervised content selection

Unsupervised content selection → clusterization (need to define distance)

salience: how informative a word is in a sentence

goal: select sentences that have more salient words

Vector Space Model (VSM)

$$A = \begin{matrix} & \mathbf{d}_1 & \mathbf{d}_2 & \mathbf{d}_3 & \\ \mathbf{t}_1 & w_{1,1} & w_{1,2} & w_{1,3} & \\ \mathbf{t}_2 & w_{2,1} & w_{2,2} & w_{2,3} & \\ \mathbf{t}_3 & w_{3,1} & w_{3,2} & w_{3,3} & \end{matrix} \quad \begin{matrix} \bullet \text{ N - \# of documents in collection} \\ \bullet \text{ M - \# of unique terms} \end{matrix}$$

$$\text{sim}(d_k, d_j) = \sum_{i=1}^M w_{i,k} \cdot w_{i,j}$$

with the distance being:

documents $d_j = (w_{1,j}, w_{2,j}, \dots, w_{M,j})$ represented as vectors of weight w .

Many methods to choose from:

- **Frequency based → TF-IDF**

combines:

$$\left. \begin{array}{l} \circ \text{ term frequency } tf_{i,j} = \frac{\# \text{ occurrences of the term } t_i \text{ in doc } d_j}{\# \text{ terms in doc } d_j} \\ \circ \text{ rarity measure } idf_i = \log \left(\frac{N}{\# \text{ docs containing } t_i} \right) \end{array} \right\} \text{ weight are defined as:}$$

$$w_{i,j} = tf_{i,j} \cdot idf_i$$

the more a word is frequent in document d_j and rare in other documents → the more it represents d_j

then the weight of the sentence is: $w_j(s_k) = \sum_{w_i \in \text{Non}_{stop}(s_k)} \frac{\text{weight}_j(w_i)}{|\text{Non}_{stop}(s_k)|}$

problem: great difference in size of the document may mislead the importance of the words.

- **Centroid based → log likelihood ratio**

think about words as pseudo-sentences → extract sentences close to such pseudo-sentence.

Discrepancy $D = -2 \log(\lambda(w_i))$: considering a word it measures the observed frequency wrt the theoretical distribution of such word in a document. The bigger D the greater the importance of the word.

- **Centrality-based methods**

computes the distances between each sentence of the current document → choose the most averagely closer sentences

Supervised content selection

Can leverage more things wrt to weight (from clustering):

- position in a sentence: at beginning or end words tend to be more important
- length of each sentence
- ...

Need a method to combine all the information → supervised model

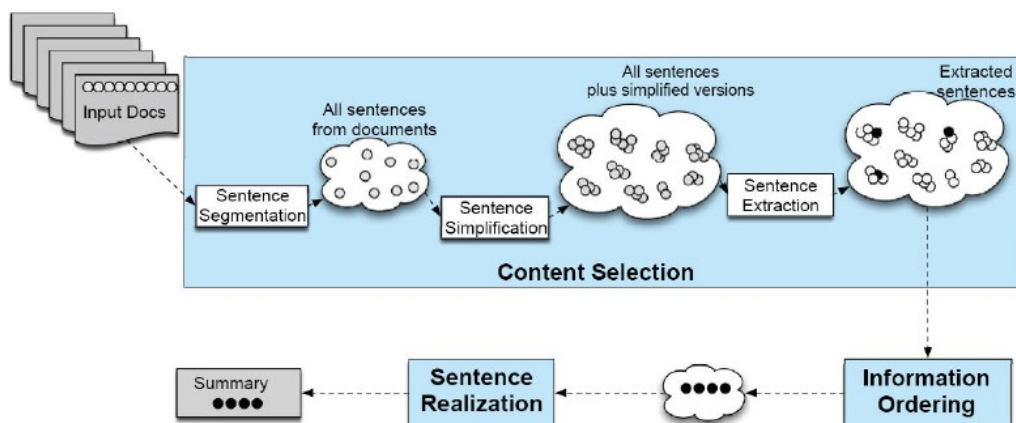
goal: classify each sentence (sample) as important or not

once we have corpus + features → can apply

usually train with pairs document/abstract summary, so sentences in the document are different from the one in the summary → need alignment between document and summary sentences (distance algorithms are used).

Multiple Documents

Summarize from multiple documents.



1. Content selection
2. Information ordering
3. Sentence realization

Content selection

Since multiple documents → redundancy which needs to be trimmed down. Selection methods:

- **Maximal Marginal Relevance (MMR)**: penalize (lowers weight) redundant terms
$$MMR_penalization_factor(s) = \lambda \cdot \max_{s_i \in summary} \text{Sim}(\underbrace{s}_{\text{current sentence}}, \underbrace{s_i}_{\text{sentence already in summary}})$$
- **Clustering**: apply a clustering algorithm to all sentences in the documents, then from each cluster select one single (centroid) sentence to put in the summary.

Information ordering

Decide how to concatenate extracted sentences. Ordering methods:

- **Chronological ordering**: sort sentences by publishing date of the documents.
Problem: lacks cohesion → solve with larger sentences chunks
- **Coherence**: in the summary put similar sentences together and different sentences far away.
- **Centering**: based on “focus”, the salient entity of each discourse segment.
Discourse coherent if focus: has certain syntactic realization (subject/object), has some transitions between this syntactic realizations (same entity is the subject of adjacent sentences).
Entity grid used for knowing where subject and objects do prefer to appear (Subject, Object, X-oblique (complement)).

Content selection + Information ordering (HMM)

- Observations: actual sentences
- Hidden states: sentence clusters, each one stands for a topic (for content selection)

HMM transition probability distribution: $P(c_j | c_i)$ which is high if c_i sentences often precedes c_j sentences. Select ordering among candidates based on highest probability that HMM offers.

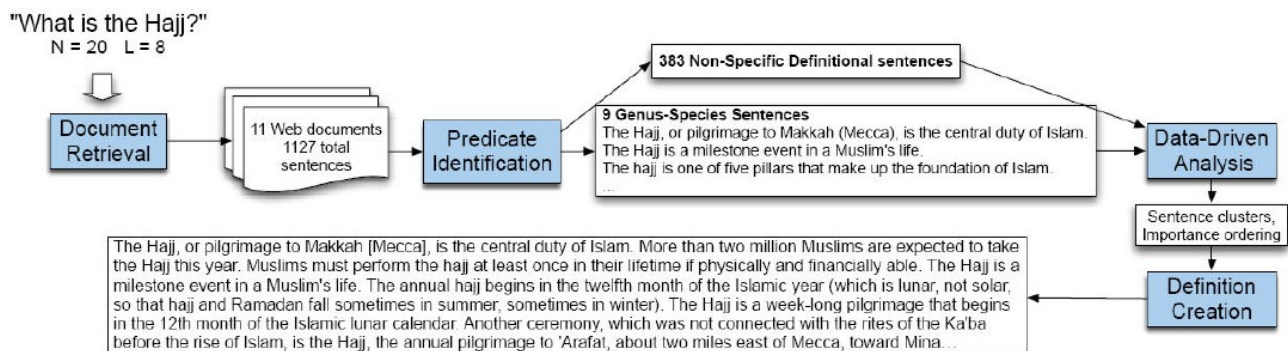
Cleaning: after ordering apply cleanup (example: full name at first mention, remove appositives after first mention).

Focused Summarization

Build answers to complex questions. Methods:

- **modify multiple-document summarization** to make use of the query (example: overlapping word with the query)
- **information extraction** methods

Information Extraction



1. Patterns are used to **identify the query class** (Definition, Biography, ...) and **extract the term** to be defined from the question
2. **Information retrieval**: engine that stores documents. Upon term request returns a ranked list of pertaining documents.
3. Sentences in the returned documents are classified and labeled based on the query class.
4. Extract information from labeled sentences and add them to the answer
5. Add to the answer some context not really pertaining to the extraction typology.

Class specific templates can be used:

<NAME> is <WHY FAMOUS>. She was born on <IRTHDATE> in <IRTHLOCATION>. She <EDUCATION>. <DESCRIPTIVE SENTENCE>. <DESCRIPTIVE SENTENCE>.

Summarization Evaluation

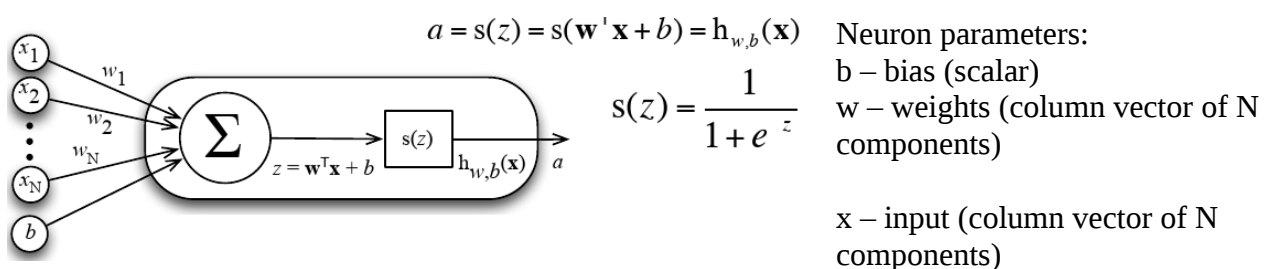
- **Extrinsic (task based)**: put the system in action
- **Intrinsic (task independent)**:
 - *Rouge*: measures overlapping n-grams from candidate summary and human generated summary
 - *Pyramid*: text subdivided in SCU (Summary Content Unit), the smallest text carrying a specific information. Then rank wrt more used SCU in multiple human-made summaries.

Neural Networks and NLP

Neural Networks

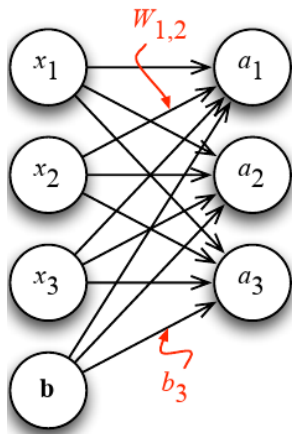
ML works better than HMM and GMM, thanks to more powerful computation devices and research advancements.

Neuron



z – scalar
 a – output (scalar)
 s – activation function

Neuron layer



$$\begin{aligned}
 a_1 &= s(W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1) \\
 a_2 &= s(W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2) \\
 a_3 &= s(W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3)
 \end{aligned}$$

$$\mathbf{x} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{a} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad \mathbf{b} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad \mathbf{W} \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix}$$

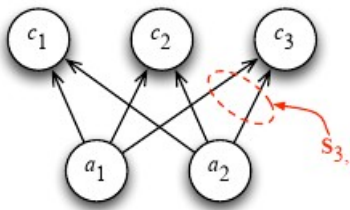
Softmax classifier

Transform a k -dimensional vector of arbitrary real values to a k -dimensional vector of real values in the range $(0,1)$ that add up to 1.

$$P(c_j | \mathbf{a}) = \frac{e^{(\mathbf{S}_{j,:})^T \mathbf{a}}}{\sum_{k=1}^K e^{(\mathbf{S}_{k,:})^T \mathbf{a}}}$$

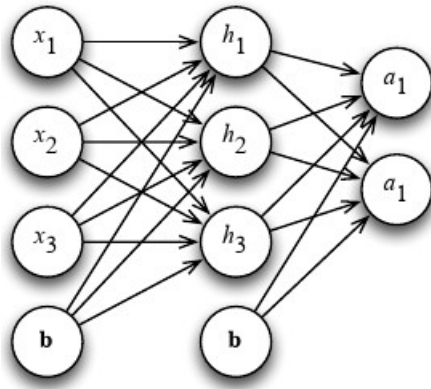
- \mathbf{a} – output (vector of real values in the range $(0,1)$ that sums up to 1)
- c_j – j class of K
- $\mathbf{S}_{j,:}$ – j row of the weight matrix

then results are the confidence in classifying the output

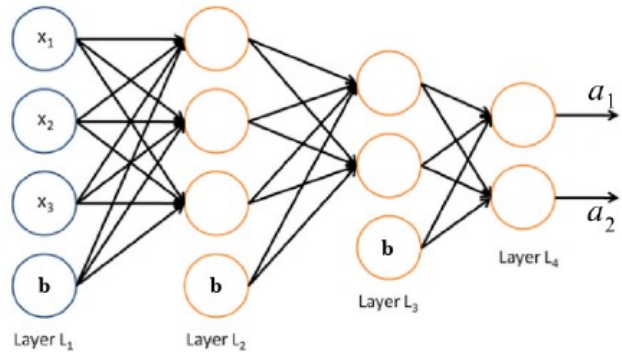


Deep neural networks

NN: input layer, hidden layer, output layer



DNN: input layer, hidden layers, output layer



Cost function

Measures how well the training of a NN performed comparing actual and correct outputs.

- Supervised learning**

Given training sample pairs $K = \{(\underbrace{x_i}_{\text{value at input layer}}, \underbrace{y_i}_{\text{correct value at output layer}}), \dots\}$

a_i actual value of the output layer.

Network learns the function $x \rightarrow a$ by minimizing the cost function (error)

Cost function to minimize $J_i(a_i, y_i)$ (example: square error)

- Unsupervised learning**

Given training samples $K = \{\underbrace{x_i}_{\text{value at input layer}}, \dots\}$

Cost function to minimize $J_i(x_i, a_i)$

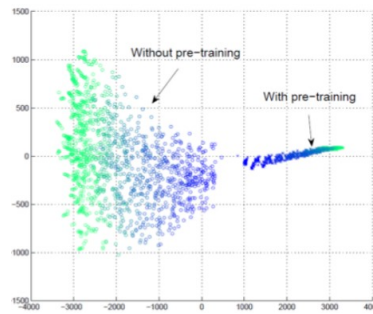
Training

2 types of learning:

- Online:** iteratively change the NN by computing the cost function
- Offline:** accumulate the errors for the whole training set and then compute the cost function

parameters initialization: random

dnn: difficult because function is not convex, possible through “effective learning” an unsupervised pre-training



error: backpropagated through the NN

Backpropagation

Goal: compute cost function wrt any weight $\frac{\sigma J}{\sigma W_{h,k}}$ and bias $\frac{\sigma J}{\sigma b_h}$ in the network

assumptions for cost function for back propagation:

- cost function as an average over the cost functions J_i for individual training examples x_i
- cost function as a function of the output a of the neural network

backpropagation steps:

1. init network parameters (random)
2. Propagation
 1. propagate input sample x_i through network. Current parameters generate output to pass to the next layer
 2. cost function J_i generate deltas δ at output layer and back-propagate through the network generating δ for all neurons
3. Weight update; for weight $w_{1 \rightarrow 2}$ connecting neuron 1 and 2:
 1. to get gradient of the weight $w_{1 \rightarrow 2}$, multiply activation a_1 and delta δ_2
 2. gradient is multiplied by the learning rate constant
 3. update $w_{1 \rightarrow 2}$ using gradient result
4. if minimum not reached goto 2

Words representation

Classic

Vocabulary represented with a vector $o_{hotel} = [0 \ 0 \ 0 \ 0 \ 1 \ 0 \]$

problem: can't relate word similarities, input length depends on vocabulary size.

Word Embedding

Represent word by means of the neighbors (window) reducing the dimensions.

Represents similarity by syntax and semantic.

$$x = \begin{bmatrix} 0.268 \\ 0.792 \\ -0.177 \\ \dots \end{bmatrix}$$

Unsupervised learning

Word in its context it's a good training sample, while a word put out of context is a bad example. Score function computes whether it's a good example or not.

1. **Word embedding matrix:** Initialize all word vectors randomly to create a lookup table.

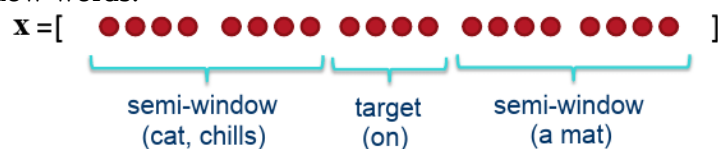
$$L = \begin{bmatrix} \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \end{bmatrix}^n$$

the cat mat ...

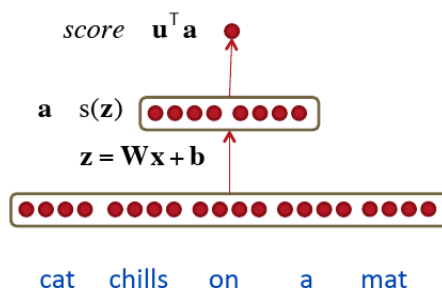
|V|

- |V| vocabulary dimension
- n dimension of the embedded space (column composed of similarity values wrt the window of the word)

2. Search in the corpus and extract the positive i-th sample.
3. Substitute a random word and generate the negative i-th sample
4. **“Embedded column vector”** $x = L \cdot \underset{O \in \{0,1\}}{O}$: vector that represents the information related to a single word with the sample embeddings. O is a one-hot column vector that selects target and window words.



5. **Score function** $score(x) = u^T \cdot a$ for $x_{+,1}$ and $x_{-,1}$ computed using the NN, with $a = s(W \cdot x_{+,i} + b)$ and $a = s(W \cdot x_{-,i} + b)$:
u: weight column vector



6. **Cost function:** $J_i = \max(0, 1 - score(x_{+,i}) + score(x_{-,i}))$ increase score of true window and lower corrupt window score.
7. **Backpropagation** and update u, W, b, x.
8. if J not minimum goto 2
9. **Update L column** corresponding to target word.

Sentence representation

Word Hashing

Technique to reduce the word's representation size.

Advantages: robust for misspelling, inflection, collisions

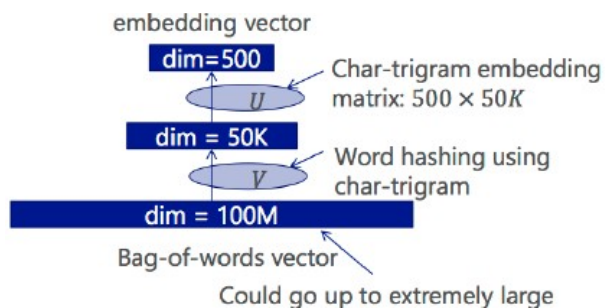
One-hot representation

$$x(cat) = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{index to word \text{"#cat\#"}}$$

Word hashing representation

$$f(cat) = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{indices to trigrams \text{"#ca"}, \text{"cat"}, \text{"at\#"}}$$

Word embedding + word hashing



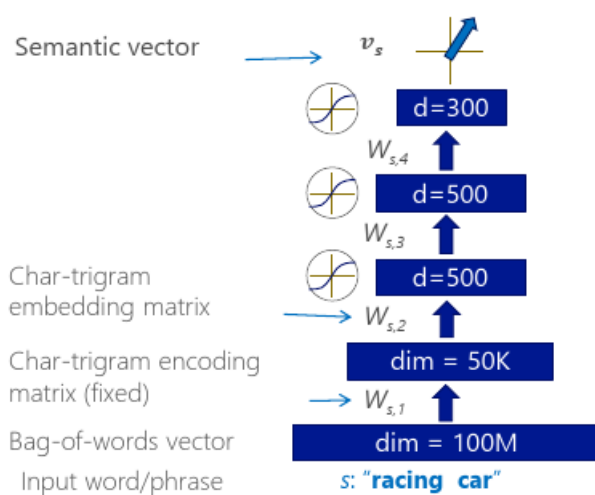
Decompose words into set of context-dependent characters.

$$\underbrace{W}_{\text{L matrix}} \rightarrow \underbrace{U}_{\text{hashed L matrix}} \times V$$

good for:

- scalability
- generalizability

DSSM: Similarity Driven Sent2Vec Model

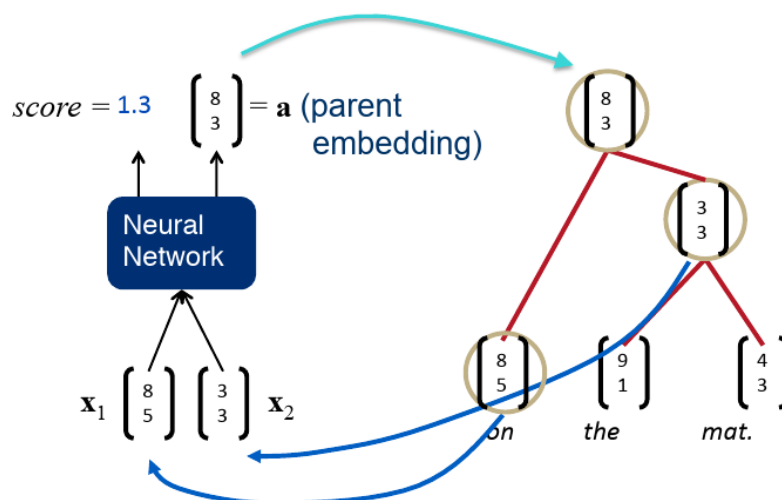


1. Initialize: random weights
2. Training: compute cosine similarity between semantic vectors and compute the gradients
3. Runtime: check similarity between semantic vectors

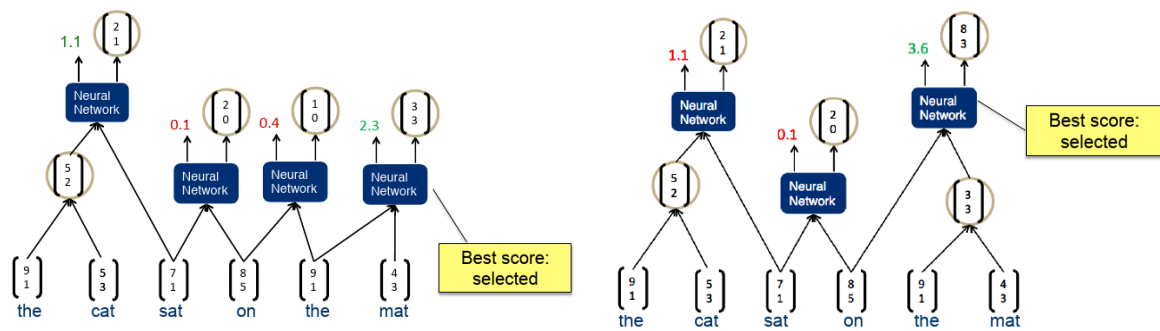
Recursive Networks

Sentence parsing

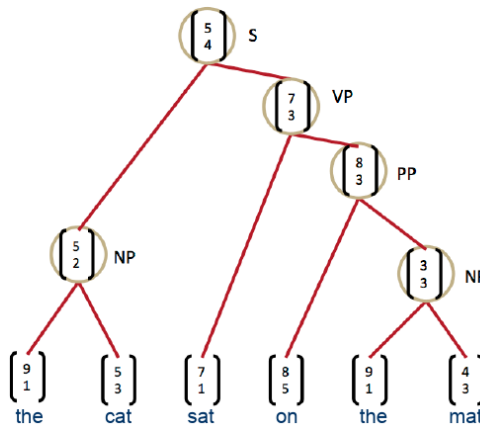
1. **Generate parse tree** starting from word embeddings.
 1. Node generation: all nodes generated with same NN.
 - inputs: 2 candidate children's representations
 - outputs:
 - *parent embedding*: embedding representation if the 2 nodes are merged
 - *score*: how plausible the new node is $a = \tanh\left(W \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b\right)$



2. Generate tree: build tree by computing scores for all possible pairs x_1 and x_2 and retain trees with best scores



2. add semantic to parse tree



Max-Margin Framework

Tree score = sum of the parsing decision scores for each node of the tree

Supervised learning uses sample pairs $(\underbrace{x_i^{(IN)}}_{\text{input sentence}}, \underbrace{y_i}_{\text{correct tree}})$

$$\text{Score function: } J_i = \sum_j^{y_i} \left(\underbrace{\text{score}(x_i^{(IN)}, y_{i,j})}_{\text{correct tree score}} - \max_{y \in A(x_i^{(IN)})} \left(\underbrace{\text{score}(x_i^{(IN)}, y)}_{\text{tentative tree score}} + \underbrace{\Delta(y, y_{i,j})}_{\text{loss: lipenalize icorrect decisions}} \right) \right)$$

$A(x_i^{(IN)})$: set of possible trees parsing sentence $x_i^{(IN)}$. Search in A performed with Greedy or Beam Search

Compositional Vector Grammars

$$\text{CVG} = \text{PCFG} + \text{Syntactically-Untied RNN}$$

(Compositional Vector Grammars) context free grammars with probabilities differentiate score function for syntactic categories

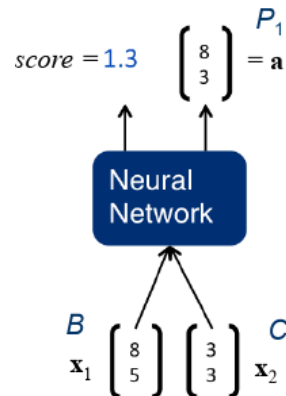
combine discrete syntactic categories with continuous semantic information

Problem: speed, each candidate score in beam search needs a matrix-vector product

Solution: compute score using a linear combination of log-likelihood from a simple PCFG+RNN

- prunes bad candidates
- Provides coarse syntactic categories of the children for each beam candidate

Score:
$$score = \left(u^{(B,C)} \right)^T a + \log P \left(P_1 \rightarrow \underbrace{BC}_{\substack{\text{tag associated with input} \\ \text{search in CFG grammar for this expansion}}} \right)$$



Language Models

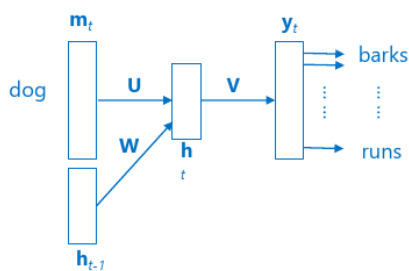
N-gram language modeling

Word only depends on n-1 preceding words

Problems:

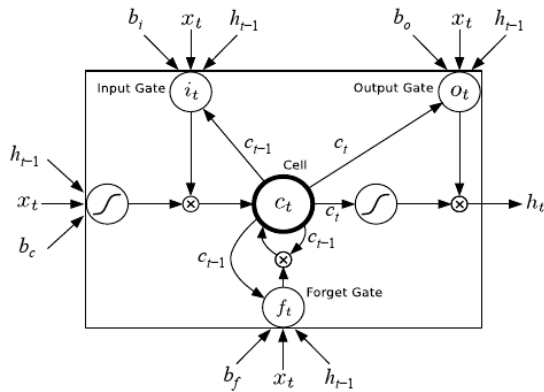
- difficult to store → lot's of parameters
- difficult to train
- cannot capture long distance dependencies → in long history becomes inaccurate

Recurrent NN for language modeling



- m_t input one-hot vector at time step t
- h_t encodes history of all words up to time step t (a vector)
- y_t distribution of output words at time step t
- by unfolding: can go back for a lot of steps, but **with depth information get lost (problem)**. h can only store limited memory and information vanishes with back propagation.

Long-short memory cell



More complex to train model, but it retains information that otherwise would be lost. Gates do manage information contained inside the LSTM:

- forget: what information to remove from the cell state
- input: new information that will be stored in cell state
- output: what to store in h

Neural Tensor Network

Allows for more interactions between input vectors wrt classic RNN.

$$v = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \cdot V^{[1:n]} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ with } v_1, \dots, v_n \text{ as scalars}$$

with:

$$a_1 = s \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \cdot V^{[1:2]} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + W \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right)$$

