Lecture Notes

On

Natural Language Processing

III B. TECH II SEMESTER (R20 AUTONOMOUS)

DEPARTMENT OF CSE

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

ACE

Engineering College

An Autonomous Institution

Ghatkesar, Hyderabad - 501 301, Telangana.

Approved by AICTE & Affiliated to JNTUH

NBA Accredited B.Tech Courses, Accorded NACC A-Grade with 3.20 CGPA

NATURAL LANGUAGE PROCESSING

B.Tech. III Year II Semester

Course Code	Category	Hours/Week			Credits	Maximum Marks		
		L	T	P	C	CIA	SEE	Total
		3	1	-	4	30	70	100
Contact Classes: 45	Tutorial Classes: 15	Practical Classes: Nil				Total Classes: 60		

Prerequisite: Data structures, finite automata and probability theory

Course Objectives:

- Introduce to some of the problems and solutions of NLP and their relation to linguistics andstatistics.

Course Outcomes:

- Show sensitivity to linguistic phenomena and an ability to model them with formal grammars.
- Understand and carry out proper experimental methodology for training and evaluatingempirical NLP systems
- Able to manipulate probabilities, construct statistical models over strings and trees, andestimate parameters using supervised and unsupervised training methods.
- Able to design, implement, and analyze NLP algorithms
- Able to design different language modeling Techniques.

Unit: I	Finding the Structure of Words:

Finding the Structure of Words: Words and Their Components, Issues and Challenges, Morphological Models

Finding the Structure of Documents: Introduction, Methods, Complexity of the Approaches, Performances of the Approaches

Unit: II	Syntax Analysis:

Syntax Analysis: Parsing Natural Language, Treebanks: A Data-Driven Approach to Syntax, Representation of Syntactic Structure, Parsing Algorithms, Models for Ambiguity Resolution in Parsing,Multilingual Issues

Unit: III	Semantic Parsing:

Semantic Parsing: Introduction, Semantic Interpretation, System Paradigms, Word SenseSystems, Software.

Unit: IV	Predicate-

Predicate-Argument Structure, Meaning Representation Systems, Software.

Unit: V	Uncertain knowledge and Learning

Discourse Processing: Cohension, Reference Resolution, Discourse Cohension and Structure

Language Modeling: Introduction, N-Gram Models, Language Model Evaluation, ParameterEstimation, Language Model Adaptation, Types of Language Models, Language-Specific ModelingProblems, Multilingual and Crosslingual Language Modeling

Text Books:

1. Multilingual natural Language Processing Applications: From Theory to Practice – Daniel M.Bikel and Imed Zitouni, Pearson Publication
2. Natural Language Processing and Information Retrieval: Tanvier Siddiqui, U.S. Tiwary

REFERENCE BOOK:

1. Speech and Natural Language Processing - Daniel Jurafsky & James H Martin, Pearson Publications

ACE
Engineering College
An AUTONOMOUS Institution
Ghatkesar, Medchal (Dist), Hyderabad, Telangana State – 501 301
(NBA Accredited B.Tech Courses Accredited NAAC with A Grade 3.20 CGPA)
Phone: 9133308533, 468, website: www.aceec.ac.in

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

S.No	Unit Number	Name of the topic	Referred Book number
1.	UNIT -I	Introduction to NLP	TB1
2.		Finding the Structure of Words: Words and Their Components	TB1
3.		Tokens, Lexemes	TB1
4.		Morphemes, Typology	TB1
5.		Issues and Challenges	TB1
6.		Morphological Models	TB2
7.		Dictionary Lookup, Finite-State Morphology	TB2
8.		Unification-Based Morphology, Functional Morphology, Morphology Induction	TB2
9.		Finding the Structure of Documents: Introduction	TB1
10.		Methods : Generative Sequence Classification Methods, Discriminative Local Classification Methods	TB1
11.		Discriminative Sequence Classification Methods, Hybrid Approaches	TB1
12.		Extensions for Global Modeling for Sentence Segmentation	TB2
13.		Complexity of the Approaches	TB1
14.		Performances of the Approaches	
15.	UNIT -II	Syntax Analysis: Introduction	TB1
16.		Parsing Natural Language	TB1
17.		Treebanks: A Data-Driven Approach to Syntax	TB1
18.		Representation of Syntactic Structure: Syntax Analysis Using Dependency Graphs	TB1
19.		Syntax Analysis Using Phrase Structure Trees	
20.		Parsing Algorithms : Shift-Reduce Parsing	TB1
21.		Hypergraphs and Chart Parsing, Minimum Spanning Trees and Dependency Parsing	TB1
22.		Models for Ambiguity Resolution in Parsing : Probabilistic Context-Free Grammars	TB1
23.		Generative Models for Parsing, Discriminative Models for Parsing	TB1
24.		Multilingual Issues	TB1
25.	UNIT-III	Semantic Parsing: Introduction	TB1
26.		Semantic Interpretation : Structural Ambiguity	TB2
27.		Word Sense, Entity and Event Resolution	TB2

28.		Predicate-Argument Structure	TB2
29.		Meaning Representation	TB1
30.		System Paradigms	TB1
31.		Word SenseSystems, Software	TB1
32.	**UNIT-IV**	Predicate - Introduction	TB1
33.		Predicate-Argument Structure	TB1
34.		Predicate-Argument Structure : Systems, Software	TB1
35.		Meaning Representation Systems	TB2
36.		Meaning Representation Systems : Systems, Software	TB2
37.		Word Sense Disambiguation	TB2
38.		Predicate-Argument Structure, Meaning Representation, Software	TB2
39.	**UNIT - V**	Discourse Processing: Cohension	TB2
40.		Reference Resolution	TB2
41.		Discourse Cohension and Structure Language Modeling: Introduction	TB2
42.		N-Gram Models, Language Model Evaluation	TB2
43.		Parameter Estimation	TB2
44.		Language Model Adaptation, Types of Language Models	TB2
45.		Language-Specific Modeling Problems	TB2
46.		Multilingual and Cross lingual Language Modeling	TB2

TEXT BOOKS :

1. Multilingual natural Language Processing Applications: From Theory to Practice – Daniel M. Bikel and Imed Zitouni, Pearson Publication
2. Natural Language Processing and Information Retrieval: Tanvier Siddiqui, U.S. Tiwary

Unit-I

- Finding the Structure of Words
- Finding the Structure of Documents
- Words and their Components
- Tokens
- Lexemes
- Morphemes
- Typology
- Issues and Challenges
- Irregularity
- Ambiguity
- Productivity
- Morphological Models
- Dictionary Lookup
- Finite-State-Morphology
- Unification-Based Morphology
- Functional Morphology
- Morphology Inductio

-Introduction

- Human Language is used to express our thoughts, and through language, we receive information and infer its meaning.
- Linguistic Expressions (words, phrases, sentences) show structure of different kinds and complexity and consist of more elementary components.
- The co-occurrence of linguistic expressions in context refines the notions they refer to in isolation and implies further meaningful relations between them.
    - The whole disciplines that look at languages from different perspectives and at different levels of detail are:

- **Morphology**- study the variable forms and functions of words.
- **Syntax**- It is concerned with the arrangement of words into phrases, clauses and sentences.
- **Phonology**- describes the word structure constraints due to pronunciation.
- **Orthography**-deals with the conventions for writing in a language.
- **Etymology** and **Lexicography**- evolution of words and explains the semantic, morphological and other links among them

Morphological Parsing

- Here we discuss about:
- How to identify words of distinct types in human languages?
- How the internal structure of words can be modelled with respect to grammatical properties and lexical concepts the words should represent?
- This discovery of word structure is called as **morphological parsing**.

Words and their Components

- Words in most languages are the smallest linguistic units that can form a complete utterance by themselves.
- Three important terms which are integral parts of a word are:
- **Phonemes** – the distinctive units of sound in spoken language.
- **Graphemes** – the smallest unit of a written language which corresponds to a phoneme.
- **Morphemes** - the minimal part of a word that delivers aspects of meaning to the word.

- Tokens
- Lexemes
- Morphemes

- Typology

Tokens

Let us look at an example in English:

Will you read the newspaper? Will you read it? I won't read it.

- Here we see two words **newspaper** and **won't.**
- In writing, newspaper and its associated concepts are very clear but in speech there are a few issues.
- When it comes to word won't linguists prefer to analyze it as two words or tokens **will** and **not**.
- This type of analysis is called **tokenization** and **normalization**.
- In Arabic or Hebrew certain tokens are concatenated in writing with the preceding or the following words, possibly changing their forms.
- This type of tokens are called clitics (I'm, we've).
- In the writing systems of Chinese, Japanese and Thai white space is not used to separate words.
- In Korean character strings are called eojeol 'word segment' and correspond to speech or cognitive units which are usually larger than words and smaller than clauses.

EXAMPLE : 학생들에게만 주셨는데

hak.sayng.tul.ey.key.man cwu.syess.nun.te[2]

haksayng-tul-eykey-man cwu-si-ess-nunte

student+*plural*+*dative*+only give+*honorific*+*past*+while

while (he/she) gave (it) only to the students

Lexemes

- There are a lot of alternative forms that can be expressed for a given word.
- Such sets are called lexemes or lexical items.
- They constitute the lexicon of a language.

- Lexemes are divided by their lexical categories such as verb, noun, adjective, adverb etc.
- The citation form of a lexeme by which it is identified is called lemma.
- In the conversion of singular mouse to plural mice we **inflect** the lexeme.
- In the case of receiver and reception we **derive** the words from the verb to receive.
- Example: Did you see him? I didn't see him? I didn't see anyone.
- Example in Czech

EXAMPLE 1–4: Viděl*as* ho? Neviděl*a* jsem ho. Neviděl*a* jsem nikoho.
saw+you-are him? not-saw I-am him. not-saw I-am no-one.

- Example in Telugu:

vAlYlYu aMxamEna wotalo neVmmaxigA naduswunnAru.

They beautiful garden slowly walking

Morphemes

- The structural components that associate the properties of word forms are called morphs.
- The morphs that by themselves represent some aspect of the meaning of a word are called **morphemes** of some function.
- Example : dis-agree-ment-s where agree is a free lexical morpheme and other elements are bound grammatical morphemes.
- Morphs when interact with each other undergo additional phonological and orthographic changes.
- These alternative forms are called allomorphs.
- Example: the past tense morphemes, plural morphemes etc.

Typology

- Morphological typology divides languages in groups. Here we outline the typology that is based on quantitative relations between words, their morphemes and their features:
- **Isolating**, or **analytic**, languages include no or relatively few words that would comprise more than one morpheme (typical members are Chinese, Vietnamese, and Thai; analytic tendencies are also found in English).
- **Synthetic** languages can combine more morphemes in one word and are further divided into agglutinative and fusional languages.
- **Agglutinative** languages have morphemes associated with only a single function at a time (as in Korean, Japanese, Finnish, and Tamil, etc.).
    - **Fusional** languages are defined by their feature-per-morpheme ratio higher than one (as in Arabic, Czech, Latin, Sanskrit, German, etc.).
        - In accordance with the notions about word formation processes mentioned earlier, we can also discern:
        - **Concatenative** languages linking morphs and morphemes one after another.
        - **Nonlinear** languages allowing structural components to merge nonsequentially to apply tonal morphemes or change the consonantal or vocalic templates of words.

Issues and Challenges

Irregularity

Ambiguity

Productivity

Issues and Challenges- Introduction

- Morphological parsing tries to remove unnecessary **irregularities** and give limits to **ambiguity** both of which exist in natural languages.
- Irregularity is all about forms and structures that are not described appropriately by a prototypical linguistic model.
- Ambiguity is indeterminacy in interpretation of expressions of language.
- In addition to ambiguity we need to deal with the issues of **syncretism**, or systematic ambiguity. (bet)
- In addition to the above morphological modelling also faces the problem of **productivity** and creativity in language, by which new but perfectly meaningful new words or new senses are coined.

IRREGULARITY

- Morphological parsing provides generalization and abstraction in the world of words.
- Irregular morphology can be seen as enforcing some extended rules the nature of which is phonological, over the underlying or prototypical regular word forms.
- In English the general past form occurs by adding –ed or –t. (accepted and built)

  The irregular verbs in English tend to take different forms in the past or in the present participle depending on the origin of the word

- A few examples:

Verb	Past Tense	Past Participle
blow	blew	blown
break	broke	broken
bring	brought	brought

Example in Arabic

whether you-saw+him? not-did I-see+him. not-did I-see anyone.

P-STEM	P–3MS	P–2FS	P–3MP	II2MS	IS1–S	IJ1–S	I-STEM	
qaraʔ	*qaraʔa*	*qaraʔti*	*qaraʔū*	*taqraʔu*	*ʔaqraʔa*	*ʔaqraʔ*	*qraʔ*	S
faʕal	*faʕal-a*	*faʕal-ti*	*faʕal-ū*	*ta-fʕal-u*	*ʔa-fʕal-a*	*ʔa-fʕal*	*fʕal*	I
faʕal	*faʕal-a*	*faʕal-ti*	*faʕal-ū*	*ta-fʕal-u*	*ʔa-fʕal-a*	*ʔa-fʕal-*	*fʕal*	M
...	...-a	...-ti	...-ū	ta-...-u	ʔa-...-a	ʔa-...-	...	
faʕā	*faʕā-a*	*faʕā-ti*	*faʕā-ū*	*ta-fā-u*	*ʔa-fā-a*	*ʔa-fā-*	*fā*	M
faʕā	*faʕā*	*faʕal-ti*	*faʕ-aw*	*ta-fā*	*ʔa-fā*	*ʔa-fa*	*fā*	I
raʔā	*raʔā*	*raʔayti*	*raʔaw*	*tarā*	*ʔarā*	*ʔara*	***rā***	S

- Roots like telus- in Telugu inflect differently in the past.

EXAMPLE

- The verb un- has two complementary 1) un- and 2) undu
- Examples

Words forms that look the same but have distinct functions or meaning are called homonyms. (Example: kind, ring, right, rose

Ambiguity

- Ambiguity is present in all aspects of morphological processing and language processing at large.
- Morphological parsing is not concerned with complete disambiguation of words in their context, however; it can effectively restrict the set of valid interpretations of a given word form.
- Example in Korean:

(-ko)		(-e)		(-un)		Meaning
묻고	***mwut.ko***	묻어	*mwut.e*	묻은	*mwut.un*	'bury'
묻고	***mwut.ko***	물어	***mwul.e***	물은	*mwul.un*	'ask'
물고	*mwul.ko*	물어	***mwul.e***	문	*mwun*	'bite'
걷고	***ket.ko***	걷어	*ket.e*	걷은	*ket.un*	'roll up'
걷고	***ket.ko***	걸어	***kel.e***	걸은	*kel.un*	'walk'
걸고	*kel.ko*	걸어	***kel.e***	건	*ken*	'hang'
굽고	***kwup.ko***	굽어	*kwup.e*	굽은	*kwup.un*	'be bent'
굽고	***kwup.ko***	구워	*kwu.we*	구운	*kwu.wun*	'bake'
이르고	***i.lu.ko***	이르러	*i.lu.le*	이른	***i.lun***	'reach'
이르고	***i.lu.ko***	일러	*il.le*	이른	***i.lun***	'say'

- The morphological disambiguation of a few languages like (Arabic) encompass not only the resolution of the structural components of words and their actual morpho-syntactic properties but also tokenization and normalization etc.
- Inverting sandhi during tokenization can provide multiple solutions to the problem of ambiguity in Indian languages. (na asatah vidyate bhavah which means the unreal has no existence)
- Example in Czech:

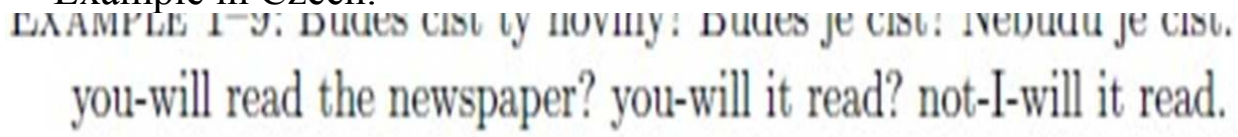
vocative, 6 locative, 7 instrumental case

	Masculine inanimate	Feminine	Feminine	Neuter
S1	dům	budova	stavba	stavení
S2	domu	budovy	stavby	stavení
S3	domu	budově	stavbě	stavení
S4	dům	budovu	stavbu	stavení
S5	dome	budovo	stavbo	stavení
S6	domu / domě	budově	stavbě	stavení
S7	domem	budovou	stavbou	stavením
P1	domy	budovy	stavby	stavení
P2	domů	budov	staveb	stavení
P3	domům	budovám	stavbám	stavením
P4	domy	budovy	stavby	stavení
P5	domy	budovy	stavby	stavení
P6	domech	budovách	stavbách	staveních
P7	domy	budovami	stavbami	staveními

Productivity

- The morphological phenomenon that some words or word classes show instances of systematic homonymy is called syncretism.
- In particular, homonymy can occur due to **neutralization** and **uninflectedness** with respect to some morpho-syntactic parameters.
- **neutralization** is about syntactic irrelevance as reflected in morphology
- **uninflectedness** is about morphology being unresponsive to a feature that is syntactically relevant.
- In English the gender category is syntactically neutralized in the case of pronouns.
- The difference between he and she, him and her are only semantic.
- An important question to be answered- Is the inventory of words in a language finite or infinite?
- In one view, language can be seen as simply a collection of utterances actually pronounced or written.
- This data set can be the linguistic corpora, a finite collection of linguistic data.

- If we consider language as a system, we discover structural devices like recursion (great-great), iteration or compounding (in-side) that allow to produce an infinite set of concrete linguistic utterances.
- This process is called as morphological productivity.
- The members of the corpus are the word types.
- The original instances of the word form is the word token.
- The distribution of words or other elements of language follow the “80/20” rule also know as the law of the vital few.
- The negation is a productive morphological operation in some languages. Examples of English are dis-, non- , un• Example in Indian Languages a-samardh
- Example in Czech:



- Let us look at an example where creativity, productivity and the issue of unknown words meet nicely.
- According to Wikipedia the word ‘googol’ is a made-up word denoting a number “one followed by hundred zeros”.
- The name of the company Google is actually a misspelled word.
- Now both of these words entered the English lexicon.

Morphological Models

- Dictionary Lookup
- Finite-State Morphology
- Unification-Based Morphology
- Functional Morphology
- Morphology Induction

Morphological Models-Introduction

- Morphological parsing is a process by which word forms of a language are associated with corresponding linguistic descriptions.
- There are many approaches to designing and implementing morphological models.
- A lot of domain specific programming languages have been created that can be very useful in implementing theoretical problems with minimal programming effort.
- There are also a few approaches that do not resort to the domain specific programming.
- We now discuss a few prominent types of computational approaches to morphology.

Dictionary Lookup

- A dictionary is a data structure that directly enables obtaining precomputed word analysis.
- Dictionaries can be implemented as lists, binary search trees, tries, hash tables etc.
- Dictionaries enumerate the set of associations between word forms and their descriptions.
- The generative power of the language is not exploited when implemented in the form of a dictionary.
- The problem with dictionary based approach is how the associated annotations are constructed and how informative and accurate they are.

Finite-State Morphology

- Finite-State morphological models are directly compiled into finitestate transducers.
- Examples of finite-state transducers are Xerox Finite-state tool (XFST) and LEX tool.
- The set of possible sequences accepted by the transducer defines the input language and the set of possible sequences emitted by the transducer defines the output language.
- In finite state computational morphology the input word forms are referred to as **surface strings** and the output strings are referred to as **lexical strings**. (the finite string children can be converted to lexical string child [+ plural])
-

Finite-State Morphology

- Let us have a relation R, and let us denote it by [∑], the set of all sequences over some set of symbols ∑ so that the domain and range of R are subsets of [∑].
- Now R is a function mapping input string to a set of output strings.

  R: [∑] ☾ {[∑]} which can be written as

  R: string ☾ {string}
- Morphological operations and processes in human languages can be expressed in finite-state terms.
- A theoretical limitation of finite state models of morphology is the problem of reduplication of words found in many natural languages.

Finite-State Morphology

- Finite-state technology can be applied to the morphological modeling of isolating and agglutinative languages very easily.
- Finite-state tools can be used to a limited extent in morphological analyzers or generators.

Unification-Based Morphology

- Unification based approaches to morphology are inspired by two things:
- The formal phase linguistic frameworks like head-driven structure grammar(HPSG).
- Languages for lexical knowledge representation like (DATR)
- The concepts and methodologies of these formalisms are closely connected to logic programming. (Prolog)
- In higher level approaches linguistic information is expressed by more appropriate data structures that can include complex values unlike finite state models.

Unification-Based Morphology

- Morphological parsing P associates linear forms Ø with alternatives of structured content **Ψ**

P: Ø ☾ {**Ψ**}

P:form ☾ {content}

- For morphological modelling word forms are best captured by regular expressions, while the linguistic content is best described through typed **feature structures** (can be viewed as directed acyclic graphs).
- Unification is the key operation by which feature structures can be merged into a more informative feature structure.

Unification-Based Morphology

• Morphological models of this kind are typically formulated as logic programs and unification is used to solve the system of constraints imposed by the model

Functional-Morphology

- Functional morphology defines its models using principles of functional programming and type theory.
- It treats morphological operations and processes as pure mathematical functions and organizes the linguistic as well as abstract elements of a model into distinct types of values and type classes.
- Functional morphology is not limited to modeling particular types of morphologies in human languages but is useful for fusional morphology.

Functional Morphology

- Functional Morphology can be used for the implementation of:
- Morphological parsing
- Morphological generation
- Lexicon browsing etc
- Along with parsing described in the previous section we can also describe inflection I, derivation D and lookup L as functions of these generic types:

I: lexeme ☾ {parameter} ☾ {form}

D: lexeme ☾ {parameter} ☾ {lexeme}

L : content ☾ {lexeme}

Morphology Induction

- Until now the focus is on finding the structure of words in diverse languages supposing we know what we are looking for.
- We now consider the problem of discovering and inducing word structure without the human insight.(unsupervised or semisupervised).
- Automated acquisition of morphological and lexical information, even if not perfect, can be reused for

bootstrapping and improving the classical morphological models, too.

- There are several challenging issues about deducing the word structure just from the forms and their context.

Finding the Structure of Documents

- Introduction
- Methods
- Complexity of the Approaches
- Performances of the Approaches
- Sentence Boundary Detection
- Topic Boundary detection

Introduction

- As we all know words form sentences.
- Sentences can be related to each other by explicit discourse connectives such as **therefore**.
- Sentences form paragraphs.
- Paragraphs are self contained units of discourse about a particular point or idea.
- Automatic extraction of structure of documents help in:
- Parsing
- Machine Translation
- Semantic Role Labelling
- Sentence boundary annotation is important for human readability of the output of the **automatic speech recognition** (ASR) system.
- Chunking the input text or speech into topically coherent blocks provides better organization and indexing of the data.
- For simplicity we consider sentence and group of sentences related to a topic as the structure elements.
- Here we discuss about two topics:

- **Sentence boundary detection:** the task of deciding where sentences start and end given a sequence of characters.
- **Topic Segmentation:** the task of determining when a topic starts and ends in a sequence of sentences.
  - Here we discuss about statistical classification approaches which base their predictions on features of the input.
  - Features of the input are local characteristics that give evidence toward the presence or absence of a sentence or a topic boundary such as:
  - Punctuation marks
  - A pause in a speech
  - A new word in a document
  - Careful design and selection of features is required in order to be successful and prevent overfitting and noise problems.
  - The statistical approaches we discuss here are language independent but every language is a challenge in itself. For example:
    - In Chinese documents words are not separated by spaces.
    - In morphologically rich languages word structure may be analyzed to extract additional features.

Such processing is usually done as a preprocessing step

Sentence Boundary Detection

- Sentence boundary detection deals with automatically segmenting a sequence of word tokens into sentence units.
- In written text in English and a few languages the beginning of a sentence usually marked with an upper case letter and the end of a sentence is marked with:
- a period(.),a question mark(?) and an exclamation mark(!)

- In addition to the role as sentence boundary markers:
- **Capitalized letters**- distinguish proper nouns
- **Periods**- used in abbreviations and numbers
- **Other punctuation marks**- used inside proper names
- Dr. can be an abbreviation for doctor or drive.
- Examples (from Brown corpus-10% of the periods are abbreviations):
- I spoke with Dr. Smith.
- My house is on Mountain Dr.
- Examples (from Wall Street Journal Corpus-47% of the periods are abbreviations):
- "This year has been difficult for both Hertz and Avis," said Charles Finnie, carrental industry analyst-yes, there is such a profession-at Alex. Brown & sons.
- An automatic method might cut the previous sentences incorrectly.
- If the preceding sentence is spoken then prosodic cues(accent, stress, rhythm, tone, pitch and intonation) mark the structure.
- One more problem of sentence segmentation in written text is spontaneously written texts (SMS and IM) have poorly used punctuation.
- If the input text comes from an automatic system (OCR or ASR) with the aim to translate images of handwritten, typewritten or printed text or spoken utterances then finding of the system boundaries must handle the errors of those systems as well.
- OCR systems confuse periods and commas and can result in meaningless sentences.

ASR transcripts lack punctuation marks and are mono-case.

- Human participants when tried to re-punctuate mono-case texts performed at an F1-measure of about 80% which shows how difficult the task is.

- Let us look at the utterance “okay no problem” in a conversation.
- It is not clear whether there is a single sentence or two in the above utterance.
- This problem is redefined for the conversational domain as the task of dialogue act segmentation.
- Dialogue acts are better defined for conversational speech using a number of markup standards.
- Dialogue Act Markup in Several Layers (DAMSL) and Meeting Recorder Dialogue Act (MRDA) are two examples of such markup standards.
- According to these standards the utterance “okay no problem” consists of two sentential units (dialogue act units) okay and no problem.

In the sentence “I think so but you should also ask him”, according to the segmentation standards there are two dialogue act tags.

- Code switch(sentences from multiple languages by multilingual speakers) is another problem that can affect the characteristics of sentences.
- Code switch also affects technical texts for which the punctuation signs can be redefined.
- Conventional rule-based sentence segmentation systems in wellformed texts rely on patterns to identify potential ends of sentences and lists of abbreviations for disambiguating them.
- Sentence segmentation can be stated as classification problem.

Topic Boundary detection

Topic segmentation is the task of automatically dividing a stream of text or speech into topically homogeneous blocks

Example:

dead. Few pictures available from the region but we do know temperatures there will be very cold tonight -7 degrees. <TOPIC_CHANGE> Peace talks expected to resume on Monday in Belfast, Northern Ireland. . . .

Figure 2–1. Example of a topic boundary in a news article

- Topic segmentation is an important task for applications like:
- Information extraction and retrieval
- Text Summarization
- In the 1990s Defense Advanced Research Projects Agency(DARPA) initiated the topic detection and Tracking (TDT) program.
- The objective was to segment a news stream into individual stories.
- Topic segmentation is a significant problem and requires a good definition of topic categories and their granularities.
- Topics are not typically flat but occur in a semantic hierarchy.
- When a statement about soccer is followed by a statement about cricket should the annotator mark a topic change??
- It is difficult to segment the text into a predefined number of topics.

In the case of TDT corpus high inter-annotator agreement (Cohen's kappa value of 0.7 to 0.9) was achieved.

- In text, topic boundaries are usually marked with distinct segmentation cues like headlines and paragraph breaks.
- Speech provides other cues such as pause duration and speaker change.

Methods

- Generative Sequence Classification Methods
- Discriminative Local Classification Methods
- Discriminative Sequence Classification Methods
- Hybrid Approach
- Extensions for Global Modeling for Sentence Segmentation

Given a boundary candidate the goal is to predict whether or not the candidate is an actual boundary

- Let $x \in X$ be the vector of features associated with a candidate and $y \in Y$ be the label predicted for that candidate.
- The label y can be b for boundary and b' for non-boundary.
- Given a set of training examples $\{x,y\}_{train}$ we need to find a function that will assign the most accurate possible label y of unseen examples xunseen.
- Sentence segmentation in text can be framed as a three class problem: sentence boundary with an abbreviation b^a, without abbreviation $b^{a'}$ and abbreviation not at boundary b^{-a}.
- In spoken language a three way classification can be made between non-boundaries b^{-1}, statement boundaries b^s and question boundaries b^q.
- For sentence or topic segmentation the problem is defined as finding the most probable sentence or topic boundaries.

- The classification can be done at each potential boundary I (**local modeling**) then the aim is to estimate the most probable boundary type y' for each candidate example xi : yi'= argmaxyi in Y P(yi/xi)
- Here y_i' is the estimated category and y_i is the possible categories.
- If we look at the candidate boundaries as a **sequence** then

$Y'=y_1',\ldots y_n'$ that have

The maximum probability given the candidate examples $X=x_1, \ldots,x_n$

$$Y'=\text{argmax}_Y P(Y/X)$$

- The methods used are categorized into:
- Local
- Sequence
- Another categorization is done according to the type of the machine learning algorithm:
- Generative
- Discriminative
- Generative sequence models estimate the joint distribution of the observations, P(X,Y) and the labels which requires specific assumptions and have good generalization properties.
- Discriminative sequence models focus on features that characterize the differences between labeling of the examples.
- These methods can be used for both sentence and topic segmentation in both text and speech.
- The only problem is that in the case of text if end-of-sentence delimiters are not included it will be difficult to categorize.
- The most commonly used generative sequence classification method for topic and sentence segmentation is the Hidden Markov Model(HMM).

$$\hat{Y} = \underset{Y}{\operatorname{argmax}}\, P(Y|X) = \underset{Y}{\operatorname{argmax}}\, \frac{P(X|Y)P(Y)}{P(X)} = \underset{Y}{\operatorname{argmax}}\, P(X|Y)P(Y)$$

- P(X) is dropped because it is the same for different Y.

Generative Sequence Classification Methods

- P(X/Y) and P(Y) can be estimated as:

$$P(X|Y) = \prod_{i-1}^{n} P(\mathbf{x}_i|y_1, \ldots, y_i)$$

$$P(Y) = \prod_{i=1}^{n} P(y_i|y_1, \ldots y_{i-1})$$

- We assume that:

$$P(\mathbf{x}_i|y_1, \ldots, y_i) \approx P(\mathbf{x}_i|y_i)$$

- A bigram model can be assumed for modeling output categories:

$$P(y_i|y_1, \ldots, y_{i-1}) \approx P(y_i|y_{i-1})$$

- The bigram case is modeled by a fully connected m-state Markov model, where m is the number of boundary categories.
- The states emit words for sentence (topic) segmentation, and the state sequence that most likely generated the word sequence is estimated.
- State transition probabilities $P(y_i/y_{i-1})$, and state observation likelihoods, $P(x_i/y_i)$, are estimated using the training data.
- The most probable boundary sequence is obtained by dynamic programming.

- Example:

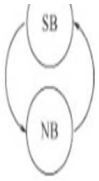


Figure 2–2. Conceptual hidden Markov model for segmentation with two states

simple two-state Markov model

Emitted Words	…	people	are	dead	few	pictures	…
State Sequence	…	NB	NB	SB	NB	NB	…

- For topic segmentation we can use n states where n is the number of topics.
- Obtaining state observation likelihoods without knowing the topic categories is the main challenge.

An imaginary token is inserted between all consecutive words in case the word preceding the boundary is not part of a disfluency.

Example 2–1: *… people NB are NB dead YB few NB pictures …*

Figure 2–3. Conceptual hidden event language model for segmentation

The extra boundary tokens can be used to capture other meta information.

The most commonly used meta information is the feedback obtained from other classifiers.

For topic segmentation the same idea can be used to model topic start and topic final sections explicitly which help for broadcast news topic segmentation.

One more extension is to capture not only words but also morphological syntactic and and other information.

Discriminative Sequence Classification Methods

- These models aim to model $P(y_i/x_i)$ directly.
- In generative approaches algorithms like naïve Bayes are used.
- In discriminative methods, discriminant functions of the feature space define the model.

- The machine learning algorithms used for discriminative classification approaches are:
- Support Vector Machines
- Boosting
- Maximum Entropy
- Regression
- Discriminative approaches have outperformed generative methods in many speech and language processing tasks.
- Training for these approaches require iterative optimization.
- In discriminative local classification each boundary is processed separately with local and contextual features.
- No global optimization (sentence or document wide) is performed unlike in sequence classification models.
- For sentence segmentation supervised learning methods have been applied.

- Transformation based learning (TBL) is used to infer rules in the supervised learning.
- The classifiers used for the task are:
- Regression trees
- Neural networks
- C 4.5 classification tree
- Maximum Entropy classifiers
- Support vector machines

- A few techniques treated sentence segmentation problem as a subtask of POS tagging by assigning a tag to punctuation similar to other tokens.
- For tagging a combination of HMM and maximum entropy approaches have been used.
- For topic segmentation a method called as TextTiling method is used.
- It uses a lexical cohesion metric in a word vector space as an indicator of topic similarity.
- TextTiling can be seen as a local classification method with single feature of similarity.

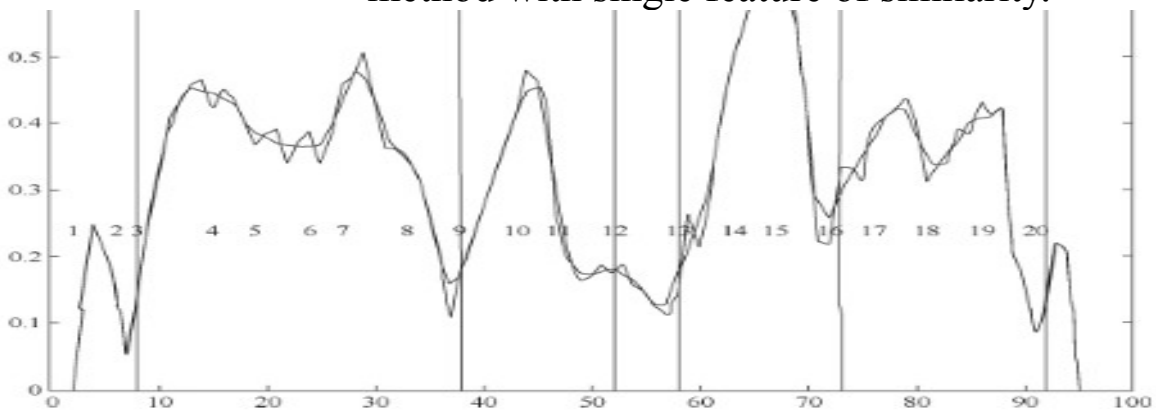


Figure 2–4. Text Tiling example (from [22])

- Two methods for computing the similarity score for topic segmentation were proposed:
- Block Comparison
- Vocabulary Introduction

- **Block comparison** compares adjacent blocks of text to see how similar they are according to how many words the adjacent blocks have in common.
- Given two blocks b_1 and b2, each having k tokens(sentences or paragraphs) the similarity score is computed by the formula:
- $W_{t,b}$ is the weight assigned to term t in block b.

- The weights may be binary or term frequency.
- The **vocabulary introduction** method assigns a score to a token-sequence gap on the basis of how many new words are seen in the interval in which it is the mid point.

$$\frac{\sum_t w_{t,b_1} \cdot w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_t w_{t,b_2}^2}}$$

- Given blocks b1 and b2 of equal number of words, w, the topical cohesion score is computed with the following formula:

$$\frac{NumNewTerms(b_1) + NumNewTerms(b_2)}{2 \times w}$$

Where NumNewTerms(b) is the number of terms in block b seen for the first time in text

$$\frac{\sum_t w_{t,b_1} \cdot w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_t w_{t,b_2}^2}}$$

- This method can be extended to exploit latent semantic analysis.
- Discriminative sequence classification methods are in general extensions of local discriminative models.
- They have additional decoding stages that find the best assignment of labels by looking at neighboring decisions to label an example.

Conditional Random Fields (CRFs) are a class of log-linear models for labelling structures

- $(Y = y_1, y_2, \ldots, y_n)$ given the sequence of feature sets extracted from the context in which they occur.
- Where $f_i(.)$ are feature functions of the observations and a clique

$$P(Y|X) \sim \frac{1}{Z(X)} \exp\left(\sum_{t=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{t-1}, y_t, y_t)\right)$$

$$Z(X) = \sum_{Y} \exp\left(\sum_{t=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{t-1}, y_t, y_t)\right)$$

  of labels and λ_i are the corresponding weights.
- Z(.) is a normalization function dependent only on the observations.

$$P(Y|X) \sim \frac{1}{Z(X)} \exp\left(\sum_{t=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{t-1}, y_t, y_t)\right)$$

$$Z(X) = \sum_{Y} \exp\left(\sum_{t=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{t-1}, y_t, y_t)\right)$$

- CRFs are trained by finding the λ parameters that maximize the likelihood of the training data, usually with a regularization term to avoid overfitting.

Hybrid Approaches

- Non-sequential discriminative classification algorithms ignore the context, which is critical for the segmentation task.
- We can add context as a feature or use CRFs which consider context.
- An alternative is to use a hybrid classification approach.

- In this approach the posterior probabilities, $P_c(y_i/x_i)$ for each candidate obtained from the classifiers such as boosting or CRF are used.
- The posterior probabilities are converted to state observation likelihoods by dividing to their priors using the Bayes rule.

Extensions for Global Modeling for Sentence Segmentation

$$\underset{y_i}{\operatorname{argmax}} \frac{P_c(y_i|\mathbf{x}_i)}{P(y_i)} = \underset{y_i}{\operatorname{argmax}}\, P(\mathbf{x}_i|y_i)$$

- Most approaches to sentence segmentation have focused on recognizing boundaries rather than sentences in themselves.
- This happened because of the number of sentence hypotheses that must be assessed in comparison to the number of boundaries.
- One approach is to segment the input according to likely sentence boundaries established by a local model.
- Train a re-ranker on the n-best lists of segmentations.
- This approach allows leveraging of sentence-level features such as scores from a syntactic parser or global prosodic features.

COMPLEXITY APPORACHES

- All the approaches have advantages and disadvantages.
- These approaches can be rated in terms of:
- Training and prediction algorithms
- Performance on real world data set
- Training of discriminative approaches is more complex than training of generative ones because they require multiple passes over the training data to adjust for their feature weights.
- Generative models such as HELMS can handle multiple orders of magnitude larger training sets and benefit from old transcripts.

$$\underset{y_i}{\operatorname{argmax}} \frac{P_c(y_i|\mathbf{x}_i)}{P(y_i)} = \underset{y_i}{\operatorname{argmax}} P(\mathbf{x}_i|y_i)$$

- They work only with a few features and do not cope well with unseen events.
- Discriminative classifiers allow for a wider variety of features and perform better on smaller training sets.
- Predicting with discriminative classifiers is slower because it is dominated by the cost of extracting more features.
- In comparison to local approaches sequence approaches have to handle additional complexity.
- They need to handle the complexity of finding the best sequence of decisions which requires evaluating all possible sequences of decisions.
- The assumption of conditional independence in generative sequence algorithms allow the use of dynamic programming to trade time for memory and decode in polynomial time.
- This complexity is measured in:
- The number of boundary candidates processed together

- The number of boundary states

  Discriminative sequence classifiers like CRFs need to repeatedly perform inference on the training data which might become expensive

Performances of the Approaches

- For sentence segmentation in speech performance is evaluated using:
- Error rate:- ratio of number of errors to the number of examples
- F1 measure
- National Institute of standards and Technology (NIST) error rate:- Number of candidates wrongly labeled divided by the number of actual boundaries.
- For sentence segmentation in text the reports on the error rate on a subset of the wall street journal corpus of about 27000 sentences are as follows:
- A typical rule-based system performs at an error rate of 1.41%
- An addition of abbreviation list lowers the error rate to 0.45%
- Combining it with a supervised classifier using POS tag features lead to an error rate of 0.31%
- An SVM based system obtains an error rate of 0.25%
- Even though the error rates seem to be very low they might effect activities like extractive summarization which depend on sentence segmentation.
- For sentence segmentation in speech reports on the Mandarin TDT4 Multilingual Broadcast News Speech Corpus are as follows:
- F1 measure of 69.1% for a MaxEnt classifier, 72.6% with Adaboost, and 72.7% with SVM using the same features.
- On a Turkish Broadcast News Corpus reports are as follows:
- F1 measure of 78.2% with HELM 86.2% with fHELM with morphology features 86.9% with Adaboost and 89.1% with CRFs.

- Reports show that on the English TDT4 broadcast news corpus Adaboost combined with HELM performs at an F1 measure 67.3%

Unit-II

Syntax

- Parsing Natural Languages
- Tree Banks: A Data Driven Approach to Syntax
- Representation of Syntactic Structure
- Parsing Algorithms
- Models for Ambiguity Resolution in Parsing
- Multilingual Issues

Syntax-Introduction

- Parsing uncovers the hidden structure of linguistic input.
- In Natural Language Applications the predicate structure of sentences can be useful.
- In NLP the syntactic analysis of the input can vary from:
- Very low level- POS tagging.
- Very high level- recovering a structural analysis that identifies the dependency between predicates and arguments in the sentence.
- The major problem in parsing natural language is the problem of ambiguity.

Parsing Natural Languages

- Let us look at the following spoken sentences:
- He wanted to go for a drive in movie.
- He wanted to go for a drive in the country.
- There is a natural pause between drive and in in the second sentence.
- This gap reflects an underlying hidden structure to the sentence.
- Parsing provides a structural description that identifies such a break in the intonation.
- Let us look at another sentence:
- The cat who lives dangerously had nine lives.
- A text-to-speech system needs to know that the first instance of the word lives is a verb and the second instance a noun.
- This is an instance of POS tagging problem.
- Another important application where parsing is important is text summarization.
- Let us look at examples for summarization:

Beyond the basic level, the operations of the three products vary widely

- The above sentence can be summarized as follows:
- The operations of the products vary.
- To do this task we first parse the first sentence.

Deleting the circled constituents PP, CD and ADVP in the previous diagram results in the short sentence

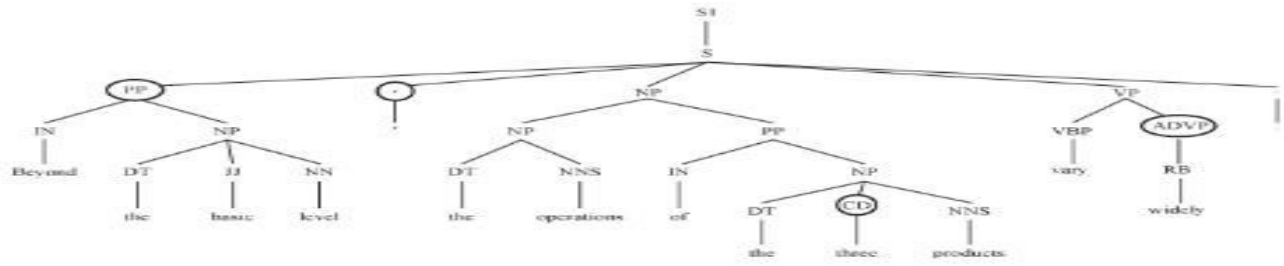


Figure 3–1. Parser output for sentence

- Let us look at another example:
- Open borders imply increasing racial fragmentation in EUROPEAN COUNTRIES.
- In the above example the capitalized phrase can be replaced with other phrases without changing the meaning of the sentence.
- Open borders imply increasing racial fragmentation in *the countries of Europe*.
- Open borders imply increasing racial fragmentation in *European states.*
- Open borders imply increasing racial fragmentation in *Europe.*
- Open borders imply increasing racial fragmentation in *European Nations*.
- Open borders imply increasing racial fragmentation in *the European countries.*
- In NLP syntactic parsing is used in many applications like:
- Statistical Machine Translation
- Information extraction from text collections
- Language summarization
- Producing entity grids for language generation
- Error correction in text
  Knowledge acquisition from language

Treebanks: A Data-Driven Approach to Syntax

- Parsing recovers information that is not explicit in the input sentence.
- Parsers require some additional knowledge beyond the input sentence that should be produced as output.
- We can write down the rules of the syntax of a sentence as a CFG.
- Here we have a CFG which represents a simple grammar of transitive verbs in English (verbs that have a subject and object noun phrase (NP), plus modifiers of verb phrases (VP) in the form of prepositional phrases (PP)).

```
S -> NP VP
NP -> 'John' | 'pockets' | D N | NP PP
VP -> V NP | VP PP
V -> 'bought'
D -> 'a'
N -> 'shirt'
PP -> P NP
P -> 'with'
```

The above CFG can produce the syntax analysis of a sentence like: John bought a shirt with pockets

- Parsing the previous sentence gives us two possible derivations.

```
(S (NP John)                    (S (NP John)
  (VP (VP (V bought)                (VP (V bought)
      (NP (D a)                   (NP (NP (D a)
        (N shirt)))                     (N shirt))
    (PP (P with)                    (PP (P with)
      (NP pockets))))                   (NP pockets)))))
```

- Writing a CFG for the syntactic analysis of natural language is problematic.

- A simple list of rules does not consider interactions between different components in the grammar.
- Listing all possible syntactic constructions in a language is a difficult task.
- It is difficult to exhaustively list lexical properties of words. This is a typical knowledge acquisition problem.
- One more problem is that the rules interact with each other in combinatorially explosive ways.
- Let us look at an example of noun phrases as a binary branching tree.
- N -> NN (Recursive Rule)
- N -> ‘natural’|’language’|’processing’|’book’
- For the input ‘natural language processing’ the recursive rules produce two ambiguous parses.

```
(N (N (N (N natural)
         (N language))
      (N processing))
   (N book))
```

```
(N (N (N natural)          (N (N natural)
      (N language))              (N (N language)
   (N processing))                  (N processing)))
```

- For CFGs it can be proved that the number of parsers obtained by using the recursive rule n times is the Catalan number (1,1,2,5,14,42, 132, 429, 1430, 4862,….) of n:

$$\mathrm{Cat}(n) = \frac{1}{n+1}\binom{2n}{n}$$

- For the input “natural language processing book” only one out of the five parsers obtained using the above CFG is correct:
- This is the second knowledge acquisition problem- We need to know not only the rules but also which analysis is most plausible for a given input sentence.
- The construction of a **tree bank** is a data driven approach to syntax analysis that allows us to address both the knowledge acquisition bottlenecks in one stroke.
- A treebank is a collection of sentences where each sentence is provided a complete syntax analysis.
- The syntax analysis for each sentence has been judged by a human expert.
- A set of annotation guidelines is written before the annotation process to ensure a consistent scheme of annotation throughout the tree bank.
- No set of syntactic rules are provided by a treebank.
- No exhaustive set of rules are assumed to exist even though assumptions about syntax are implicit in a treebank.
- The consistency of syntax analysis in a treebank is measured using interannotator agreement by having approximately 10% overlapped material annotated by more than one annotator.
- Treebanks provide annotations of syntactic structure for a large sample of sentences.
- A supervised machine learning method can be used to train the parser.
- Treebanks solve the first knowledge acquisition problem of finding the grammar underlying the syntax analysis because the analysis is directly given instead of a grammar.
- The second problem of knowledge acquisition is also solved by treebanks.
- Each sentence in a treebank has ben given its most plausible syntactic analysis.

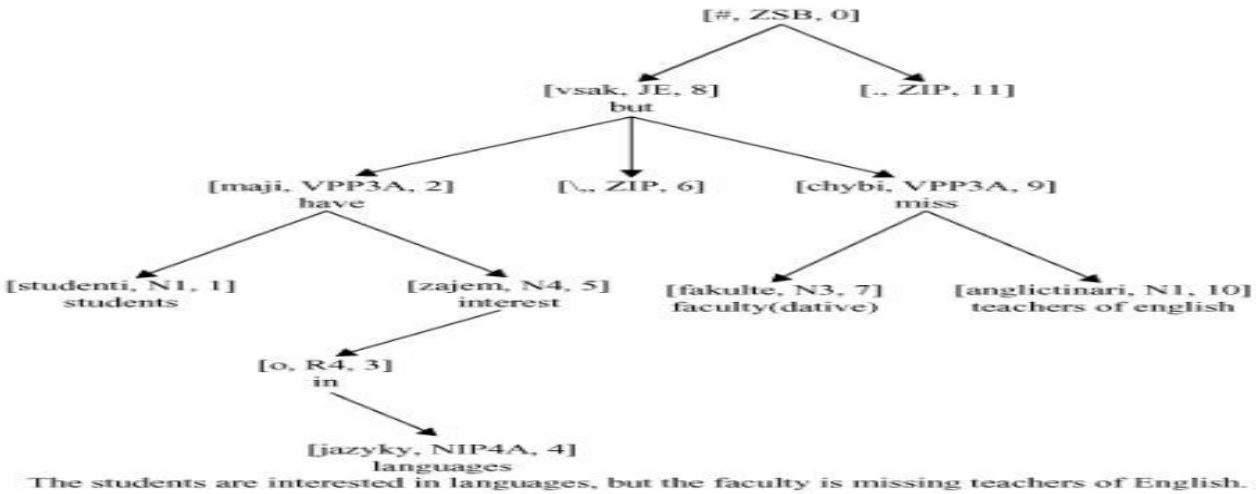
- Supervised learning algorithms can be used to learn a scoring function over all possible syntax analyses.
- For real time data the parser uses the scoring function to return the syntax analysis that has the highest score.
- Two main approaches to syntax analysis that are used to construct treebanks are:
- Dependency graphs
- Phase structure trees
- These two representations are very closely connected to each other and under some assumptions one can be converted to the other.
- Dependency analysis is used for free word order languages like Indian languages.
- Phrase structure analysis is used to provide additional information about long distance dependencies for languages like English and French.
- In the discussion to follow we examine three main components for building a parser:
- **The representation of syntactic structure**- it involves the use of a varying amount of linguistic knowledge to build a treebank.
- **The training and decoding algorithms**- they deal with the potentially exponential search space.
- **Methods to model ambiguity**- provides a way to rank parses to recover the most likely parse.

Representation of Syntactic Structure

- Syntax Analysis using dependency graphs
- Syntax Analysis using phrase structure trees

- **Syntax Analysis using dependency graphs**
- In dependency graphs the head of a phrase is connected with the dependents in that phrase using directed connections.
- The head-dependent relationship can be semantic (head-modifier) or syntactic (head-specifier).
- The main difference between dependency graphs and phrase structure trees is that dependency analysis make minimal assumptions about syntactic structure.
- Dependency graphs treat the words in the input sentence as the only vertices in the graph which are linked together by directed arcs representing syntactic dependencies.
- One typical definition of dependency graph is as follows:
- In dependency syntactic parsing the task is to derive a syntactic structure for an input sentence by identifying the syntactic head of each word in the sentence.
- The nodes are the words of the input sentence and the arcs are the binary relations from head to dependent.
- It is often assumed that all words except one have a syntactic head.
- It means that the graph will be a tree with the single independent node as the root.
- In labeled dependency parsing the parser assigns a specific type to each dependency relation holding between head word and dependent word.

In the current discussion we will be discussing about dependency trees only where each word depends on exactly one parent either another word or a dummy .



In dependency trees the 0 index is used to indicate the root symbol and the directed arcs are drawn from head word to the dependent wordIn the figure in the previous slide [fakulte,N3,7] is the seventh word in the sentence with POS tag N3 and it has dative case.

- Here is a textual representation of a labeled dependency tree:

1	They	PRP	2	SBJ
2	persuaded	VBD	0	ROOT
3	Mr.	NNP	4	NMOD
4	Trotter	NNP	2	IOBJ
5	to	TO	6	VMOD
6	take	VB	2	OBJ
7	it	PRP	6	OBJ
8	back	RB	6	PRT
9	.	.	2	P

An important notion in dependency analysis is the notion of **projectivity**

- A projective dependency tree is one where we put the words in a linear order based on the sentence with the root symbol in the first position.
- The dependency arcs are then drawn above the words without any crossing dependencies.
- Example:

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Figure 14.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

- Let us look at an example where a sentence contains an extra position to the right of a noun phrase modifier phrase which requires a crossing dependency.

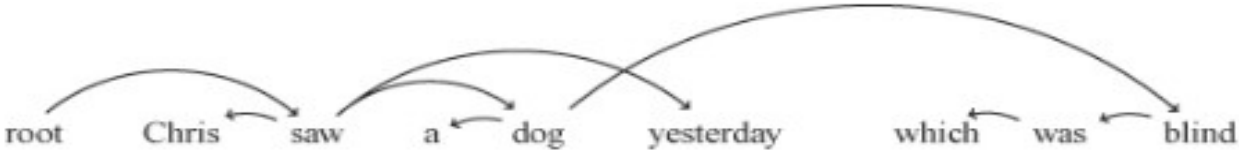


Figure 3–3. An unlabeled nonprojective dependency tree with a crossing dependency

- English has very few cases in a treebank that needs such a non projective analysis.
- In languages like Czech, Turkish, Telugu the number of non productive dependencies are much higher.
- Let us look at a multilingual comparison of crossing dependencies across a few languages:

	Ar	Ba	Ca	Ch	Cz	En	Gr	Hu	It	Tu
% deps	0.4	2.9	0.1	0.0	1.9	0.3	1.1	2.9	0.5	5.5
% sents	10.1	26.2	2.9	0.0	23.2	6.7	20.3	26.4	7.4	33.3

- Ar=Arabic;Ba=Basque;Ca=Catalan;Ch=Chinese;Cz=Czech;En =English; Gr=Greek;Hu=Hungarian;It=Italian;Tu=Turkish
- Dependency graphs in treebanks do not explicitly distinguish between projective and non-projective dependency tree analyses.
- Parsing algorithms are sometimes forced to distinguish between projective and non-projective dependencies.
- Let us try to setup dependency links in a CFG.

X0_2 -> X0_1* X2_1

X0_1 -> x0*

X2_1 -> X1_1 X2_2*

X1_1 -> x1*

X2_2 -> X2_3* X3_1

X2_3 -> x2*

X3_1 -> x3*

- In the CFG the terminal symbols are x0,x1,x2,x3.
- The asterisk picks out a single symbol in the right hand side of each rule that specifies the dependency link.
- We can look at the asterisk as either a separate annotation on the nonterminal or simply as a new nonterminal in the probabilistic context-free grammar(PCFG).
-

- The dependency tree equivalent to the preceding CFG is as follows:

- If we can convert a dependency tree into an equivalent CFG then the dependency tree must be projective.
- In a CFG converted from a dependency tree we have only the following three types of rules:
- One type of rule to introduce the terminal symbol
- Two rules where Y is dependent on X or vice-versa.
- The head word of X or Y can be traced by following the asterisk symbol.

$$Z \rightarrow X^* Y$$

$$Z \rightarrow X Y^*$$

$$A \rightarrow a^*$$

Syntax Analysis using Dependency Graphs

- Now let us look at an example non-projective dependency tree:

- When we convert this dependency tree to a CFG with * notation it can only capture the fact that X3 depends on X2 or X1 depends on X3.

```
X2_3 -> X1_1 X2_2*
X1_1 -> x1
X2_2 -> X2_1* X3_1
X2_1 -> x2
X3_1 -> x3
```

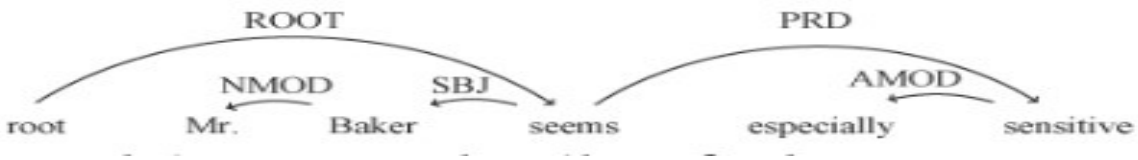
- There is no CFG that can capture the non-projective dependency.
- Projective dependency can be defined as follows:
- For each word in the sentence its descendants form a contiguous substring of the sentence.
- Non-Projectivity can be defined as follows:
- A non-projective dependency means that there is a word in the sentence such that its descendants do not form a contiguous substring of the sentence.
- In non-projective dependency there is a non-terminal Z such that Z derives spans(x_i,x_k) and (x_{k+p},x_j) for some p>0.
  - It means there must be a rule Z->PQ where P derives (x_1,x_k) and Q derives (x_{k+p},x_j).
  - By the definition of CFG this is valid if p=0 because P and Q must be continuous substrings.
  - Hence non-projective dependency can not be converted to a CFG.

Syntax Analysis using phrase structure trees

- A phrase structure syntax analysis can be viewed as implicitly having a predicate argument structure associated with it.
- Now let us look at an example sentence:
- Mr. Baker seems especially sensitive

```
(S (NP-SBJ (NNP Mr.)
           (NNP Baker))
   (VP (VBZ seems)
       (ADJP-PRD (RB especially)
                 (JJ sensitive))))
```

Predicate-argument structure:
seems((especially(sensitive))(Mr. Baker))



- We can find some similarity between Phase Structure Trees and Dependency Trees.
- We now look at some examples of Phase Structure Analysis in tree banks and see how null elements are used to localize certain predicateargument dependencies in the tree structure.
- In this example we can see that an NP dominates a trace *T* which is a null element (same as ∈).

- The empty trace has an index and is associated with the WHNP (WHnoun phrase) constituent with the same index.
- This co-indexing allows us to infer the predicate-argument structure shown in the tree.

  In this example “the ball” is actually not the logical subject of the predicate.

```
(SBARQ (WHNP-1 What)
    (SQ is (NP-SBJ Tim)
        (VP eating (NP *T*-1)))
    ?)

Predicate-argument structure:
eat(Tim, what)
```

- It has been displaced due to the passive construction.

  “Chris” the actual subject is marked as LGS (Logical subjects in passives) enabling the recovery of predicate argument structure for the sentence.

```
(VP was (VP thrown)
    (NP *-1)
    (PP by (NP-LGS Chris))))

Predicate-argument structure:
throw(Chris, the ball)
```

- In this example we can see that different syntactic phenomena are combined in the corpus.

```
(SBARQ (WHNP-1 Who)
       (SQ was (NP-SBJ-2 *T*-1)
               (VP believed (S (NP-SBJ-3 *-2)
                               (VP to (VP have
                                          (VP been
                                              (VP shot
                                                  (NP *-3)))))))
       ?)

Predicate-argument structure:
believe(*someone*, shoot(*someone*, who))
```

- Both the analysis are combined to provide the predicate argument structure in such cases.
- Here we see a pair of examples to show how null elements are used to annotate the presence of a subject for a predicate even if it is not explicit in the sentence.
- In the first case the analysis marks the missing subject for “take back” as the object of the verb *persuaded*.

- In the second case the missing subject for “take back ” is the subject of the verb *promised*.
- The dependency analysis for “persuaded” and “promised” do not make such a distinction.

  The dependency analysis for the two sentences is as follows:

```
(S (NP-SBJ (PRP They))
  (VP (VP (VBD persuaded)
      (NP-1 (NNP Mr.)
        (NNP Trotter))
      (S (NP-SBJ (-NONE- *-1))
        (VP (TO to)
          (VP (VB take)
            (NP (PRP it))
            (PRT (RB back))))))))
```

Predicate argument structure:

persuade(they, Mr. Trotter take_back(Mr. Trotter it))

```
(S (NP-SBJ-1 (PRP They))
  (VP (VP (VBD promised)
      (NP (NNP Mr.)
        (NNP Trotter))
      (S (NP-SBJ (-NONE- *-1))
        (VP (TO to)
          (VP (VB take)
            (NP (PRP it))
            (PRT (RB back))))))))
```

Predicate argument structure:

promise(they, Mr. Trotter, take_back(they, it))

```
1 They       PRP 2 SBJ
2 persuaded  VBD 0 ROOT
3 Mr.        NNP 4 NMOD
4 Trotter    NNP 2 IOBJ
5 to         TO  6 VMOD
6 take       VB  2 OBJ
7 it         PRP 6 OBJ
8 back       RB  6 PRT
9 .          .   2 P
```

```
1 They       PRP 2 SBJ
2 promised   VBD 0 ROOT
3 Mr.        NNP 4 NMOD
4 Trotter    NNP 2 IOBJ
5 to         TO  6 VMOD
6 take       VB  2 OBJ
7 it         PRP 6 OBJ
8 back       RB  6 PRT
9 .          .   2 P
```

Most statistical parsers trained using Phrase Structure treebanks ignore these differences

- Here we look at one example from the Chinese treebank which uses IP instead of S.
- This is a move from transformational grammar-based phrase structure to government-binding (GB) based phrase structure.
- It is very difficult to take a CFG-based parser initially developed for English parsing and adapt it to Chinese parsing by training it on Chinese phrase structure treebank.

```
                    (CP (IP (NP-SBJ (-NONE- *T*-2))
                            (VP (VA 新/new)))
                        (DEC 的)))
                (NP (NN 核销/verification and cancellation)
                    (NN 制度/system))))
    (VP (PP-LOC (P 在/in)
                (NP-PN (NR 西藏/Tibet)))
        (ADVP (AD 全面/fully))
        (VP (VV 实施/operating))))

English translation:
A (foreign exchange) settlement and sale system and a verification and
cancellation system that is newly created is fully operational in Tibet.
```

Parsing Algorithms

- An interesting property of rightmost derivation is revealed if we arrange the derivation in reverse order.

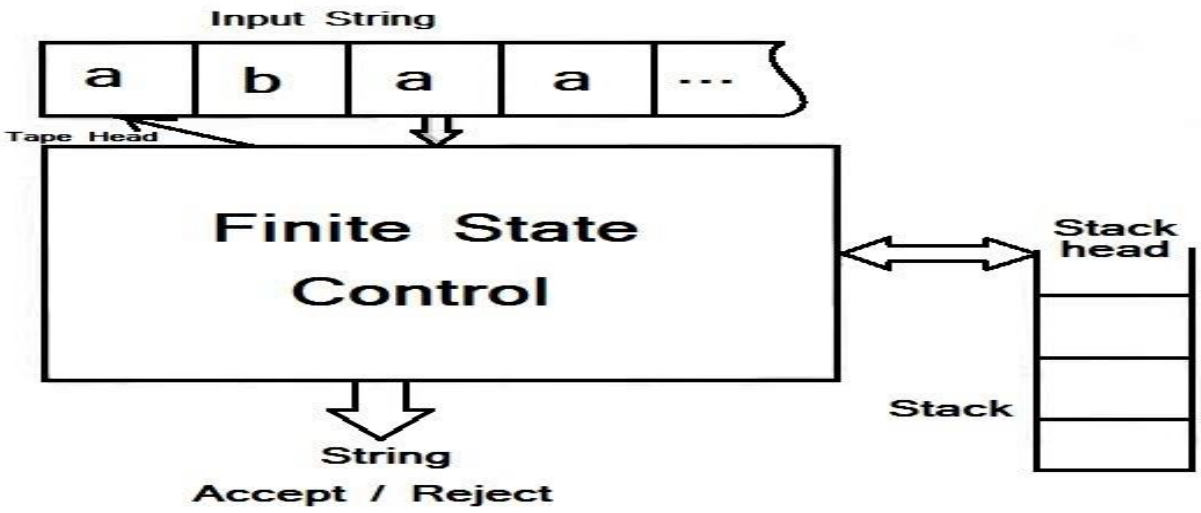
```
'a and b or c'
=> N 'and b or c'        # use rule N -> a
=> N 'and' N 'or c'      # use rule N -> b
=> N 'or c'              # use rule N -> N and N
=> N 'or' N              # use rule N -> c
=> N                     # use rule N -> N or N
```

- The above sequence corresponds to the construction of the above parse tree from left to right, one symbol at a time.
- There can be many parse trees for the given input string.

Another rightmost derivation for the given input string is as follows

SHIFT -REDUCING PARSING

- Every CFG has an automaton equivalent to it called the pushdown automaton.
- The shift-reduce parsing algorithm is defined as follows:



1. Start with an empty stack and the buffer contains the input string.
2. Exit with success if the top of the stack contains the start symbol of the grammar and if the buffer is empty.
3. Choose between the following two steps (if the choice is ambiguous, choose one based on an oracle):
   - Shift a symbol from the buffer onto the stack.
   - If the top k symbols of the stack are $\alpha_1 \ldots \alpha_k$ which corresponds to the right-hand side of a CFG rule $A \to \alpha_1 \ldots \alpha_k$ then replace the top k symbols with the left-hand side non-terminal A.
4. Exit with failure if no action can be taken in previous step.
5. Else, go to Step 2.

- For the example “a and b or c” the parsing steps are s follows:

a	a	and b or c	shift a
(N a)	N	and b or c	reduce N -> a
(N a) and	N and	b or c	shift and
(N a) and b	N and b	or c	shift b
(N a) and (N b)	N and N	or c	reduce N -> b
(N (N a) and (N b))	N	or c	reduce N -> a
(N (N a) and (N b)) or	N or	c	shift or
(N (N a) and (N b)) or c	N or c		shift c
(N (N a) and (N b)) or (N c)	N or N		reduce N -> c
(N (N (N a) and (N b)) or (N c))	N		reduce N -> N or N
(N (N (N a) and (N b)) or (N c))	N		Accept!

HYPER GRAPHS AND CHART PARSING

Dependency tree	Stack	Input	Action
root	root	a and b or c	Init
root a	root a	and b or c	shift a
root a and	root a and	b or c	shift and
root a and	root and	b or c	a ← and
root a and b	root and b	or c	shift b
root a and b	root and	or c	and → b
root a and b or	root and or	c	shift or
root a and b or	root or	c	and ← or
root a and b or c	root or c		shift c
root a and b or c	root or		or → c
root a and b or c	root		root → or

- Shift-reduce parsing allows a linear time parser but requires access to an oracle.

- CFGs in the worst case need backtracking and have a worst case parsing algorithm which run in $O(n^3)$ where n is the size of the input.
- Variants of this algorithm are used in statistical parsers that attempt to search the space of possible parse trees without the limitation of left-to-right parsing.

Our example CFG G is rewritten as new CFG G_c which contains up to two non-terminals on the right hand side

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

```
N -> N N^
N^ -> 'and' N
N -> N Nv
Nv -> 'or' N
N -> 'a' | 'b' | 'c'
```

We can specialize the CFG Gc to a particular input string by creating a new CFG that represents all possible parse trees that are valid in grammar G_c for this particular input sentence.

- For the input "a and b or c" the new CFG Cf that represents the forest of parse trees can be constructed.

Let the input string be broken up into spans 0 a 1 and 2 b 3 or 4 c 5.

```
N^[1,3] -> 'and'[1,2] N[2,3]
N^[1,5] -> 'and'[1,2] N[2,5]
N[0,5] -> N[0,3] Nv[3,5]
N[2,5] -> N[2,3] Nv[3,5]
Nv[3,5] -> 'or'[3,4] N[4,5]
N[0,1] -> 'a'[0,1]
N[2,3] -> 'b'[2,3]
N[4,5] -> 'c'[4,5]
```

- Here a parsing algorithm is defined as taking as input a CFG and an input string and producing a specialized CFG that represents all legal parsers for the input.
- A parser has to create all the valid specialized rules from the start symbol nonterminal that spans the entire string to the leaf nodes that are the input tokens.

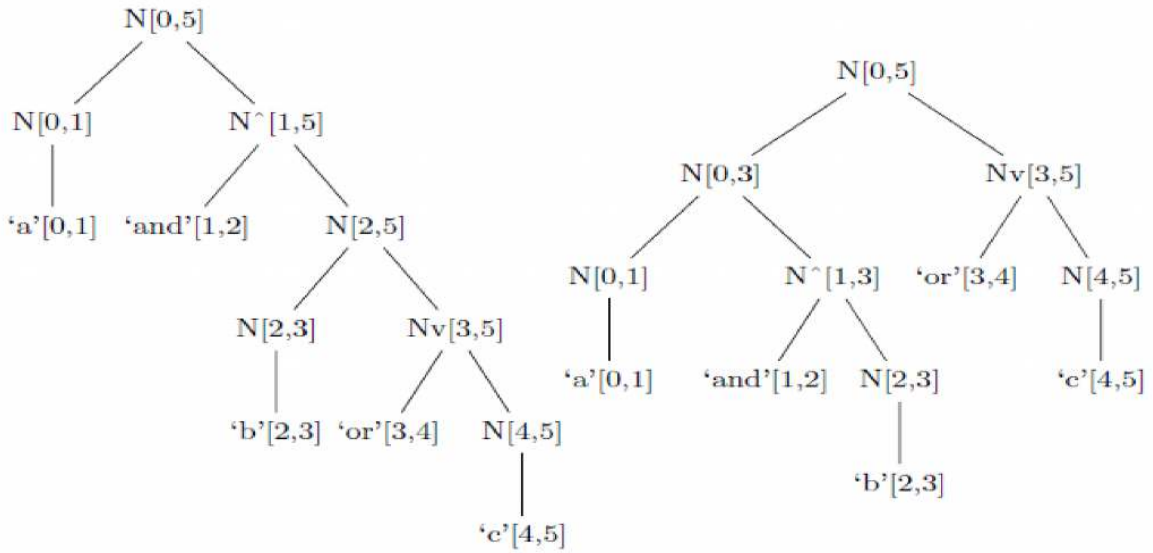


Figure 4: Parse trees embedded in the *specialized* CFG for a particular input string. The nodes with the same label, e.g. *N[0,5]*, *N[0,1]*, *and[1,2]*, *N[2,3]*, and *Nv[3,5]* can be merged to form a hypergraph representation of all parses for the input.

- Now let us look at the steps the parser has to take to construct a specialized CFG.
- Let us consider the rules that generate only lexical items:

N[0,1] -> 'a'[0,1]

N[2,3] -> 'b'[2,3]

N[4,5] -> 'c'[4,5]

Minimum Spanning Trees and Dependency Parsing:

Dependency parsing is a type of syntactic parsing that represents the grammaticalstructure of a sentence as a directed acyclic graph (DAG). The nodes of the graphrepresent the words of the sentence, and the edges represent the syntactic relationships between the words.

Minimum spanning tree (MST) algorithms are often used for dependency parsing, asthey provide an efficient way to find the most likely parse for a sentence given a set

of syntactic dependencies.Here's an example of how a MST algorithm can be used for dependency parsing:
Consider the sentence "The cat chased the mouse". We can represent this sentenceas a graph with nodes for each word and edges representing the syntacticdependencies between them:

We can use a MST algorithm to find the most likely parse for this graph. One popularalgorithm for this is the Chu-Liu/Edmonds algorithm:

1. We first remove all self-loops and multiple edges in the graph. This is becausea valid dependency tree must be acyclic and have only one edge between any two nodes.
2. We then choose a node to be the root of the tree. In this example, we can
choose "chased" to be the root since it is the main verb of the sentence.
3. We then compute the scores for each edge in the graph based on a scoring
function that takes into account the probability of each edge being a valid
dependency. The score function can be based on various linguistic features,
such as part-of-speech tags or word embeddings.

4. We use the MST algorithm to find the tree that maximizes the total score of its edges. The MST algorithm starts with a set of edges that connect the root node to each of its immediate dependents, and iteratively adds edges that connect other nodes to the tree. At each iteration, we select the edge with the highest score that does not create a cycle in the tree.
5. Once the MST algorithm has constructed the tree, we can assign a label to each edge in the tree based on the type of dependency it represents (e.g.,subject, object, etc.).

The resulting dependency tree for the example sentence is shown below:
In this tree, each node represents a word in the sentence, and each edge represents a syntacticdependency between two words. Dependency parsing can be useful for many NLP tasks, such as information extraction, machine translation, and sentiment analysis.One advantage of dependency parsing is that it captures more fine-grained syntactic information
than phrase-structure parsing, as represents the relationships between individual words rather
than just the hierarchical structure of phrases. However, dependency parsing can be more difficult to perform accurately than phrase-structure parsing, as it requires more sophisticated algorithms and models to capture the nuances of syntactic dependencies.

5.Models for Ambiguity Resolution in Parsing:

Ambiguity resolution is an important problem in natural language processing (NLP) as many sentences can have multiple valid syntactic parses. This means that the same sentence can be represented by multiple phrase structure trees or dependency graphs. Resolving ambiguity is crucial for many NLP applications, such as machine
translation, text-to-speech synthesis, and information retrieval.
Here are some common models for ambiguity resolution in parsing:

1. Rule-based models: Rule-based models use hand-crafted grammars and rules to disambiguate sentences. These rules can be based on linguistic knowledgeor heuristics, and can help resolve ambiguity by preferring certain syntactic structures over others. For example, a rule-based model might prefer a noun phrase followed by a verb phrase as the primary syntactic structure for a given sentence.

2. Statistical models: Statistical models use machine learning algorithms to
learn from large corpora of text and make predictions about the most likely

syntactic structure for a given sentence. These models can be based on various features, such as part-of-speech tags, word embeddings, or contextual information. For example, a statistical model might learn to associate certain word sequences with specific syntactic structures.

3. **Hybrid models**: Hybrid models combine both rule-based and statistical approaches to resolve ambiguity. These models can use rules to guide the parsing process and statistical models to make more fine-grained predictions.

For example, a hybrid model might use a rule-based approach to identify the main syntactic structures in a sentence, and then use a statistical model to disambiguate specific substructures.

4. **Neural network models**: Neural network models use deep learning techniques to learn from large amounts of text and make predictions about the most likely syntactic structure for a given sentence. These models can be based on various neural architectures, such as recurrent neural networks (RNNs) or transformer models. For example, a neural network model might use an attention mechanism to learn which words in a sentence are most relevant for predicting the syntactic structure.

5. **Ensemble models**: Ensemble models combine the predictions of multiple parsing models to achieve higher accuracy and robustness. These models can be based on various techniques, such as voting, weighting, or stacking.

For example, an ensemble model might combine the predictions of a rule-based model, a statistical model, and a neural network model to improve the overall accuracy of the parsing system.

Overall, there are many models for ambiguity resolution in parsing, each with its own strengths and weaknesses. The choice of model

depends on the specific application and the available resources, such as training data and computational power.

5.1 Probabilistic Context-Free Grammars :

Probabilistic context-free grammars (PCFGs) are a popular model for ambiguity resolution in parsing. PCFGs extend context-free grammars (CFGs) by assigning probabilities to each production rule, representing the likelihood of generating a certain symbol given its parent symbol.

PCFGs can be used to compute the probability of a parse tree for a given sentence,which can then be used to select the most likely parse. The probability of a parse tree is computed by multiplying the probabilities of its constituent production rules, from the root symbol down to the leaves. The probability of a sentence is computed by summing the probabilities of all parse trees that generate the sentence.

Here is an example of a PCFG for the sentence "the cat saw the dog":

S -> NP VP [1.0]
NP -> Det N [0.6]
NP -> N [0.4]
VP -> V NP [0.8]
VP -> V [0.2]
Det -> "the" [0.9]
Det -> "a" [0.1]
N -> "cat" [0.5]
N -> "dog" [0.5]
V -> "saw" [1.0]

In this PCFG, each production rule is annotated with a probability. For example, therule NP -> Det N [0.6] has a probability of 0.6, indicating that a noun phrase can be generated by first generating a determiner, followed by a noun, with a probability of 0.6.

To parse the sentence "the cat saw the dog" using this PCFG, we can use the CKY algorithm to generate all possible parse trees and compute their probabilities.

Thealgorithm starts by filling in the table of all possible subtrees for each span of the sentence, and then combines these subtrees using the production rules of the PCFG.

The final cell in the table represents the probability of the best parse tree for the
entire sentence.

Using the probabilities from the PCFG, the CKY algorithm generates the followingparse tree for the sentence "the cat saw the dog":

```
S
/ \
NP VP
/ \ / \
Det N V NP
| | | / \
the cat saw the dog
```

The probability of this parse tree is computed as follows:

P(S -> NP VP) * P(NP -> Det N) * P(Det -> "the") * P(N -> "cat") * P(VP -> V NP) * P(V ->"saw") * P(NP -> Det N) * P(Det -> "the") * P(N -> "dog") = 1.0 * 0.6 * 0.9 * 0.5 * 0.8 *
1.0 * 0.6 * 0.9 * 0.5 = 0.11664

Thus, the probability of the best parse tree for the sentence "the cat saw the dog" is 0.11664. This probability can be used to select the most likely parse among all possible parse trees for the sentence.

5.2 Generative Models for Parsing:

Generative models for parsing are a family of models that generate a sentence's parse tree by generating each node in the tree according to a set of probabilistic rules. One such model is the probabilistic earley parser.

The earley parser uses a chart data structure to store all possible parse trees for a sentence. The parser starts with an empty chart, and then adds new parse trees to the chart as it progresses through the sentence. The parser consists of three main stages: prediction, scanning, and completion.

In the prediction stage, the parser generates new items in the chart by applying grammar rules that can generate non-terminal symbols. For

example, if the grammar has a rule S -> NP VP, the parser would predict the presence of an S symbol in the
current span of the sentence by adding a new item to the chart that indicates that an S symbol can be generated by an NP symbol followed by a VP symbol.
In the scanning stage, the parser checks whether a word in the sentence can be
assigned to a non-terminal symbol in the chart. For example, if the parser has predicted an NP symbol in the current span of the sentence, and the word "dog"appears in that span, the parser would add a new item to the chart that indicates that the NP symbol can be generated by the word "dog".In the completion stage, the parser c0ombines items in the chart that have the same end position and can be combined according to the grammar rules. For example, if the parser has added an item to the chart that indicates that an NP symbol can be generated by the word "dog", and another item that indicates that a VP symbol can be generated by the word "saw" and an NP symbol, the parser would add a new item it the chart that indicates that an S symbol can be generated by an NP symbolfollowed by a VP symbol.

Here is an example of a probabilistic earley parser applied to the sentence "the cat saw the dog":

Grammar:

S -> NP VP [1.0]
NP -> Det N [0.6]
NP -> N [0.4]
VP -> V NP [0.8]
VP -> V [0.2]
Det -> "the" [0.9]
Det -> "a" [0.1]
N -> "cat" [0.5]
N -> "dog" [0.5]
V -> "saw" [1.0]
Initial chart:
0: [S -> * NP VP [1.0], 0, 0]

0: [NP -> * Det N [0.6], 0, 0]
0: [NP -> * N [0.4], 0, 0]
0: [VP -> * V NP [0.8], 0, 0]
0: [VP -> * V [0.2], 0, 0]
0: [Det -> * "the" [0.9], 0, 0]
0: [Det -> * "a" [0.1], 0, 0]
0: [N -> * "cat" [0.5], 0, 0]
0: [N -> * "dog" [0.5], 0, 0]
0: [V -> * "saw" [1.0], 0, 0]
Predicting S:
0: [S -> * NP VP [1.0], 0, 0]
1: [NP -> * Det N [0.6], 0, 0]
1: [NP -> * N [0.4], 0, 0]
1: [VP -> * V NP [0.8], 0

5.3 Discriminative Models for Parsing:

Discriminative models for parsing are a family of models that predict a sentence'sparse tree by learning to discriminate between different possible trees. One such model is the maximum entropy markov model.

The maximum entropy markov model (MEMM) is a discriminative model that models the conditional probability of a parse tree given a sentence. The model is trained on a corpus of labeled sentences and their corresponding parse trees. During training, the model learns a set of feature functions that map the current state of the parser (i.e., the current span of the sentence and the partial parse tree constructed so far) to a set of binary features that are indicative of a particular parse tree. The model then learns the weight of each feature function using maximum likelihood estimation.

During testing, the MEMM uses the learned feature functions and weights to score each possible parse tree for the input sentence. The model then selects the parse tree with the highest score as the final parse tree for the sentence.

Here is an example of a MEMM applied to the sentence "the cat saw the dog":

Features:

F1: current word is "the"
F2: current word is "cat"
F3: current word is "saw"
F4: current word is "dog"
F5: current span is "the cat"
F6: current span is "cat saw"
F7: current span is "saw the"
F8: current span is "the dog"
F9: partial parse tree is "S -> NP VP"

Weights:

F1: 1.2
F2: 0.5
F3: 0.9
F4: 1.1
F5: 0.8
F6: 0.6
F7: 0.7
F8: 0.9
F9: 1.5

Possible parse trees and their scores:

S -> NP VP
- NP -> Det N
- - Det -> "the"
- - N -> "cat"
- VP -> V NP
- - V -> "saw"
- - NP -> Det N
- - - Det -> "the"
- - - N -> "dog"

Score: 5.7

S -> NP VP

- NP -> N
- - N -> "cat"
- VP -> V NP
- - V -> "saw"
- - NP -> Det N
- - - Det -> "the"
- - - N -> "dog"
Score: 4.9
S -> NP VP
- NP -> Det N
- - Det -> "the"
- - N -> "cat"
- VP -> V
- - V -> "saw"
- NP -> Det N
- - Det -> "the"
- - N -> "dog"
Score: 3.5
Selected parse tree:
S -> NP VP
- NP -> Det N
- - Det -> "the"
- - N -> "cat"
- VP -> V NP
- - V -> "saw"
- - NP -> Det N

- - - Det -> "the"
- - - N -> "dog"
Score: 5.7

In this example, the MEMM generates a score for each possible parse tree and
selects the parse tree with the highest score as the final parse tree for the sentence.

The selected parse tree corresponds to the correct parse for the sentence.

6.**Multilingual Issues:**

In natural language processing (NLP), a token is a sequence of characters that represents a single unit of meaning. In other words, it is a word or a piece of a word that has a specific meaning within a language. The process of splitting a text into individual tokens is called tokenization.

However, the definition of what constitutes a token can vary depending on the language being analyzed. This is because different languages have different rules for how words are constructed, how they are written, and how they are used in context.

For example, in English, words are typically separated by spaces, making it relatively easy to tokenize a sentence into individual words. However, in some languages, such as Chinese or Japanese, there are no spaces between words, and the text must be segmented into individual units of meaning based on other cues, such as syntax or context.Furthermore, even within a single language, there can be variation in how words are spelled or written. For example, in English, words can be spelled with or without hyphens or apostrophes, and there can be differences in spelling between American English and British English.

Multilingual issues in tokenization arisebecause different languages can have different character sets, which means that the same sequence of characters can represent different words in different languages. Additionally, some languages have complex morphology, which means that a single word can have many different forms that represent different grammatical features or meanings.

To address these issues, NLP researchers have developed multilingual tokenization

techniques that take into account the specific linguistic features of different
languages. These techniques can include using language-specific dictionaries,models, or rules to identify the boundaries between words or units of meaning in different languages.

6.1 **Tokenization**, Case, and Encoding:

Tokenization, case, and encoding are all important aspects of natural language
processing (NLP) that are used to preprocess text data before it can be analyzed by machine learning algorithms. Here are some examples of each:

Tokenization:

Tokenization is the process of splitting a text into individual tokens or words. In English, this is typically done by splitting the text on whitespace and punctuation marks. For example, the sentence "The quick brown fox jumps over the lazy dog."would be tokenized into the following list of words:

1. ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."].

Case:2. Case refers to the use of upper and lower case letters in text. In NLP, it is often important to standardize the case of words to avoid treating the same word as different simply because it appears in different case. For example, the words "apple" and "Apple" should be treated as the same word.

Encoding:

3. Encoding refers to the process of representing text data in a way that can be processed by machine learning algorithms. One common encoding method used in NLP is Unicode, which is a character encoding standard that can represent a wide range of characters from different languages.

Here is an example of how tokenization, case, and encoding might be applied to asentence of text:

Text: "The quick brown fox jumps over the lazy dog."

Tokenization: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]
Case: ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]
Encoding: [0x74, 0x68, 0x65, 0x20, 0x71, 0x75, 0x69, 0x63, 0x6b, 0x20, 0x62, 0x72,0x6f, 0x77, 0x6e, 0x20, 0x66, 0x6f, 0x78, 0x20, 0x6a, 0x75, 0x6d, 0x70, 0x73, 0x20,0x6f, 0x76, 0x65, 0x72, 0x20, 0x74, 0x68, 0x65, 0x20, 0x6c, 0x61, 0x7a, 0x79, 0x20, 0x64, 0x6f, 0x67, 0x2e]

Note that the encoding is represented in hexadecimal to show the underlying by test that represent the text.

6.2 Word Segmentation:

Word segmentation is one of the most basic tasks in Natural Language Processing(NLP), and it involves identifying the boundaries between words in a sentence.

However, in some languages, such as Chinese and Japanese, there is no clear spacing or punctuation between words, which makes word segmentation more challenging.In Chinese, for example, a sentence like "我喜欢中文" (which means "I like Chinese")could be segmented in different ways, such as "我 / 喜欢 / 中文" or "我喜欢 / 中文".Similarly, in Japanese, a sentence like "私は日本語が好きです" (which also means "I like Japanese") could be segmented in different ways, such as "私は / 日本語が / 好きです" or "私は日本語 / が好きです".

Here are some examples of the challenges of word segmentation in different languages:

- Chinese: In addition to the lack of spacing between words, Chinese also has a large number of homophones, which are words that sound the same but have different meanings. For example, the words "你" (you) and "年" (year) sound the same in Mandarin, but they are written with different characters.

● Japanese: Japanese also has a large number of homophones, but it also has
different writing systems, including kanji (Chinese characters), hiragana, and katakana. Kanji can often have multiple readings, which makes word
segmentation more complex.
● Thai: Thai has no spaces between words, and it also has no capitalization or punctuation. In addition, Thai has a unique script with many consonants that can be combined with different vowel signs to form words.
● Vietnamese: Vietnamese uses the Latin alphabet, but it also has many
diacritics (accent marks) that can change the meaning of a word. In addition,Vietnamese words can be formed by combining smaller words, which makes
word segmentation more complex.

To address these challenges, NLP researchers have developed various techniques for word segmentation, including rule-based approaches, statistical models, and neural networks. However, word segmentation is still an active area of research,
especially for low-resource languages where large amounts of annotated data are not available.

6.3 Morphology:
Morphology is the study of the structure of words and how they are formed from
smaller units called morphemes. Morphological analysis is important in many natural language processing tasks, such as machine translation and speech
recognition, because it helps to identify the underlying structure of words and to disambiguate their meanings.

Here are some examples of the challenges of morphology in different languages:

● Turkish: Turkish has a rich morphology, with a complex system of affixes that can be added to words to convey different meanings. For example, the word
"kitap" (book) can be modified with different suffixes to indicate things like
possession, plurality, or tense.
● Arabic: Arabic also has a rich morphology, with a complex system of prefixes,
suffixes, and infixes that can be added to words to convey different meanings.
For example, the root "k-t-b" (meaning "write") can be modified with different affixes to form words like "kitab" (book) and "kataba" (he wrote).
● Finnish: Finnish has a complex morphology, with a large number of cases, suffixes, and vowel harmony rules that can affect the form of a word. For example, the word "käsi" (hand) can be modified with different suffixes to indicate things like possession, location, or movement.
● Swahili: Swahili has a complex morphology, with a large number of prefixes and suffixes that can be added to words to convey different meanings. Fore xample, the word "kutaka" (to want) can be modified with different prefixes and suffixes to indicate things like tense, negation, or subject agreement.
To address these challenges, NLP researchers have developed various techniquesfor morphological analysis, including rule-based approaches, statistical models, andneural networks. However, morphological analysis is still an active area of research,especially for low-resource languages where large amounts of annotated data are not available.

UNIT -3

Semantic Parsing: Introduction, Semantic Interpretation, System Paradigms, Word Sense Systems, Software.

Semantic Parsing

- Semantics is the study of meaning and parsing is the examination of the meaning in a minute way.
- Semantic parsing is the process of identifying meaning chunks contained in an information signal in an attempt to transform it so

that it can be manipulated by a computer program to perform higher level tasks.

- In the current case the information signal is human language text.
- The term semantic parsing has been used by researchers to represent various levels of granularity of meaning representations.

INRODUCTION

- Research in language understanding is the identification of a meaning representation that is detailed enough to allow reasoning systems to make deductions.
- It is general enough that it can be used across many domains with little to no adaptation.
- Two approaches have emerged in natural language processing for language understanding.
- In the first approach a specific rich meaning representation is created for a limited domain for use by applications that are restricted to that domain.
- Example: air travel reservations, football game simulations, querying a geographic database.
- In the second approach a related set of intermediate meaning representations is created from low level, to midlevel and the bigger understanding task is divided into multiple smaller pieces that are more manageable such as word sense disambiguation.
- In the first approach the meaning representations are tied to a specific domain.
- In the second approach the meaning representations cover the overall meaning.
- We do not yet have a detailed overall representation that would cover across domains.
- Here we treat the world as though it has exactly two types of meaning representations:
- A domain dependent deeper representation (deep semantic parsing)

- A set of relatively shallow but general purpose low level and intermediate representations. (shallow semantic parsing)

The first approach reusability of the representation across domains is very limited

- In the second approach it is difficult to construct a general purpose ontology and create symbols that are shallow enough to be learnt but detailed enough to be useful for all possible applications.

Now the community has moved from the more detailed domain dependent representation to more shallow ones

Semantic Interpretation

- Structural Ambiguity
- Word Sense
- Entity and Event Resolution
- Predicate-Argument structure
- Meaning Representation

Semantic parsing is considered as a part of a large process **semantic interpretation**

Semantic interpretation is a kind of representation of text that can be fed into a computer to allow further computational manipulations and search.

- Here we discuss about some of the main components of this process.

- We begin the discuss with **Syntactic structures** by Chomsky which introduced the concept of a transformational phrase structure grammar.

Later Katz and Fodor wrote a paper "The structure of a semantic theory" where they proposed a few properties a semantic theory should possesses. A semantic theory should be able to :

- Explain sentences having ambiguous meaning. (Example: the sentence "the bill is large" can represent money or the beak of a bird)
- Resolve the ambiguities of the words in the context.(Example: the sentence "the bill is large but need not be paid" can be disambiguated)
- Identify meaningless but syntactically well-formed sentences. (Example "colorless green ideas sleep furiously")
- Identify syntactically or transformationally unrelated paraphrases of a concept having the same semantic content.
- We now look at some requirements for achieving a semantic representation.

Structural Ambiguity

- When we talk of structure, we refer to the syntactic structure of sentences.
- Since syntax and semantics have such strong interaction most theories of semantic interpretation refer to the underlying syntactic representation.

Syntax has become the first stage of processing followed by various other stages in the process of semantic interpretation

Word Sense

- In any given language the same word type or word lemma is used in different contexts and with different morphological variants to represent different entities or concepts in the world.
- For example the word nail represents a part of the human anatomy and also to represent an iron object.
- Humans can easily identify the use of nail in the following sentences:
- He nailed the loose arm of the chair with a hammer.
- He bought a box of nails from the hardware store.
- He went to the beauty salon to get his nails clipped
- He went to get a manicure. His nails had grown very long.
- Resolving the sense of words in a discourse, therefore, constitutes one of the steps in the process of semantic interpretation.

Entity and Event Resolution

- Any discourse consists of a set of entities participating in a series of explicit or implicit events over a period of time.
- The next important component of semantic interpretation is the identification of various entities that are sprinkled across the discourse using the same or different phrases.
- Two predominant tasks have become popular over the years:

Named entity recognition

- Coreference resolution

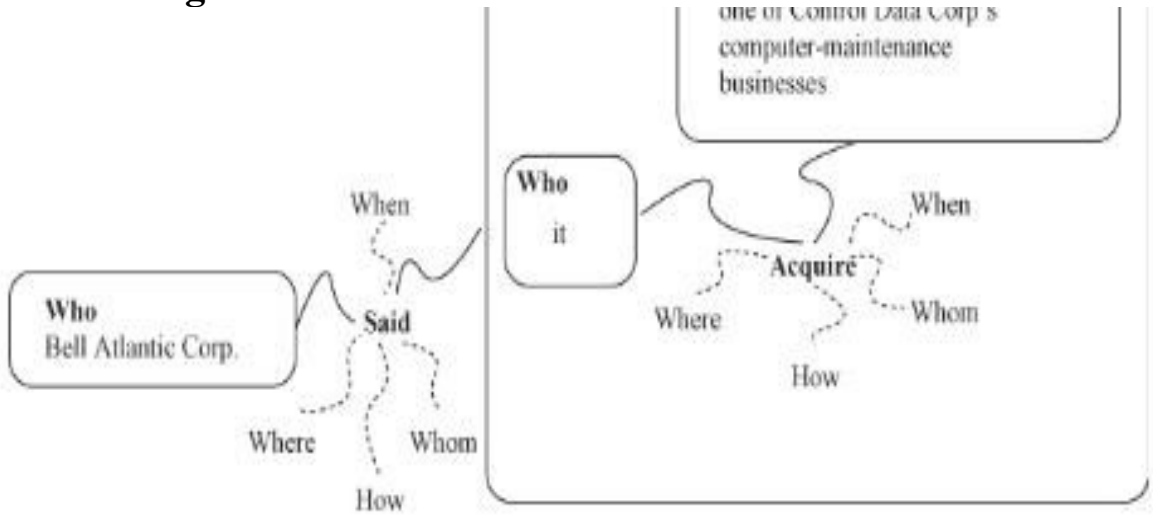
Predicate-Argument Structure

- Another level of semantic structure is identifying the participants of the entities in these events.

- Resolving the argument structure of the predicates in a sentence is where we identify which entities play what part in which event.

This process can be defined as the identification of who did what to whom, when where why and how

Predicate-Argument Structure



Meaning Representation

- The final process of the semantic interpretation is to build a semantic representation or meaning representation that can be manipulated by algorithms to various application ends.
- This process is called the deep representation.

- The following examples show sample sentences and their meaning representation for the RoboCup and GeoQuery domains:
- If our player 2 has the ball, then position our player 5 in the midfield. ((bowner(player our 2)) (do(player our 5) (pos(midfield)))).
- Which river is the longest? answer(x1,longest(x1 river(x1)))
- This is a domain-specific approach. Our further discussion here will be about domain independent approaches.

System Paradigms

- Researchers from linguistics community have examined meaning representations at different levels of granularity and generality exploring the space of numerous languages.
- Many of the experimental conditions do not have hand annotated data.
- The approaches of semantic interpretation generally fall into the following three categories:
- **System Architectures**
- **Knowledge based:** These systems use a predefined set of rules or a knowledge base to obtain a solution to a new problem.
- **Unsupervised:** These systems tend to require minimal human intervention to be functional by using existing resources that can be bootstrapped for a particular application or problem domain.

System Paradigms

- **Supervised:** These systems involve manual annotation of some phenomena that appear in a sufficient quantity of data so that machine learning algorithms can be applied. Feature functions are created to allow each problem instance to be projected into a space of features.

Semi-supervised: In instances where human annotation is difficult we can use machine generated output or bootstrap an existing model by having humans correct its output.

- **Scope**
- **Domain Dependent:** These systems are specific to certain domains such as air travel reservations or simulated football coaching.
- **Domain Independent:** These systems are general enough that the techniques can be applicable to multiple domains without little or no change.
- **Coverage**
- **Shallow:** These systems tend to produce an intermediate representation that can then be converted to one that a machine can base its actions on.
- **Deep:** These systems usually create a terminal representation that is directly consumed by a machine or application.

Word Sense

- **Resources**
- **Systems**
- **Software**
- In a compositional approach the semantics is composed of the meaning of its parts in a discourse.
- The smallest parts in textual discourse are the words themselves:
- Tokens that appear in the text
- Lemmatized parts of the tokens
- It is not clear whether it is possible to identify a finite set of senses that each word in a language exhibits in various contexts.
- Attempts to solve this problem range from rule based, knowledge based to completely unsupervised, supervised and semi-supervised learning.
- The early systems were either rule based or knowledge based and used dictionary definitions of senses of words.
- Unsupervised word sense disambiguation techniques induce the sense of a word as it appears in various corpora.

- Supervised approaches for word sense disambiguation assume that a word can evoke only one sense in a given context.
- The coarser the granularity of senses for a word the more consistent the annotation and more learnable they become.
- There is an increased chance that this lower granularity might not identify nuances that are fine enough for the consuming application.
- The number of applications which need word sense disambiguation are few and for that reason very few manually sense-tagged text corpora have been produced.
- The absence of standard criteria has also prevented the merging of various resources that have sense information.
- In information retrieval it is an accepted fact that multiple words in a query matching with multiple words in the document provide an implicit disambiguation.
- In speech recognition also context classes have proven to be applicable than word classes.
- In the case of domain specific applications words generally map to a unique concept and therefore no need for word sense disambiguation.
- The reasons for the lack of progress in word sense disambiguation are:
- Lack of standardized evaluations
- The range of resources needed to provide the required knowledge as compared to other tasks
- The difficulty of obtaining adequately large sense-tagged data sets.
- Techniques used to measure the performance of an automatic word sense disambiguation system is an important issue.
- One proposal for performance measure is called most frequent sense (MFS) baseline.
- A very important property of a gold standard sense-tagged corpus is that it should be replicable (multiple annotators accepting it) to a high degree.

- Word sense ambiguities can be of three principal types:
- Homonymy- words share the same spelling but different meaning (bank).
- Polysemy- fine nuances can be assigned to the word depending on the context (financial bank and bank of clouds).

Categorical ambiguity- the word book can be a noun or a verb

- In English word senses have been annotated for each POS separately.
- In a few languages like Chinese sense annotation is done per lemma and ranges across all POS because distinction between a noun and a verb is not clear.
- The availability of resources is the key factor in the disambiguation of word senses in corpora.
- There is no hand-tagged sense data available until recently.

RESOURCES

- Early work on word sense disambiguation used machine readable dictionaries or thesaurus as knowledge sources.
- Prominent sources were:
- Longman dictionary of contemporary English
- Roget's thesaurus
- Later a significant lexicographical resource WordNet was created.
- WordNet in addition to being a lexicographical resource has a rich taxonomy connecting words across many different relationships such as :

- Hypernymy- a broader class of things (bird)
- Hyponymy- a subclass of hypernymy (crow,pegion etc)
- Homonymy- words share the same spelling but different meaning (bank)
- Meronymy- it denotes a constituent of something (finger is a meronymy of hand)

- WordNet in addition to being used for sense disambiguation can also be used to create a semantic concordance (SEMCOR of Brown Corpus).

- Other corpuses that were developed for English language are:
- DSO corpus of sense-tagged English- It was created by tagging WordNet version 1.5 senses on the Brown and Wall Street Journal Corpara for 121 nouns and 70 verbs.
- Onto Notes corpus- It is released through the Linguistic Data Consortium (LDC). It has tagged a significant number of verb (2,700) and noun (2200) lemmas covering lemmas.
- Cyc- It is a good example of a useful resource that creates a representation of common sense knowledge about objects and events.
- Efforts for creating resources for other languages are as follows:
- HowNet- a network of words for Chinese

The Global WordNet association keeps track of WordNet development across various languages.

- A few semiautomatic methods were used for expanding coverage were used for languages like Greek.

Systems

- Rule based
- Supervised

- Unsupervised
- Algorithms motivated by Cross-linguistic Evidence
  - ➢ Semi-Supervised

Let us discuss about some sense disambiguation systems.

We can classify the sense disambiguation system into four main categories:

- Rule based or knowledge based
- Supervised
- Unsupervised

Rule based

- Semi-supervised
- The first generation of word sense disambiguation systems were primarily based on dictionary sense definitions.
- Access to the exact rules and the systems was very limited.
- Here we look at a few techniques and algorithms which are accessible.
- The first generation word sense disambiguation algorithms were mostly based on computerized dictionaries.
- We now have a look at Lesk algorithm which can be used as a baseline for comparing word sense disambiguation performance.

```
4: for all sense ∈ senses of word do
5:    signature ← set of words in gloss and examples of sense
6:    overlap ← COMPUTEOVERLAP(signature, context)
7:    if overlap gt max-overlap then
8:       max-overlap ← overlap
9:       best-sense ← sense
10:   end if
11: end for
12: return best-sense
```

- • A few modifications can be made to the Lesk algorithm so that synonyms, hypernyms, hyponyms, meromynys, and so on of the words in the context as well as in the dictionary definition are used to get accurate overlap statistic.
- One more algorithm for dictionary-based word sense algorithm used Roget's Thesaurus categories. (Yarowsky)
- It was able to classify unseen words into 1042 categories using the corpus 10-million-word Grolier's Encyclopedia.
- The method consists of three steps:
- The first step is a collection of contexts
- The second step computes weights for each of the salient words.
- The amount of context used was 50 words on each side of the target word.

- P(w/Rcat) is the probability of a word w occurring in the context of a Roget's category Rcat.
- In the third step, the unseen words in the test set are classified into the category that has the maximum weight.

$$\frac{P(w_i|RCat)}{P(w_i)}$$

3. Use the weights for predicting the appropriate category of the word in the test corpus.

$$\arg\max_{RCat} \sum_{w} \log \frac{P(w_i|RCat)P(RCat)}{P(w_i)}$$

Figure 4–2. Algorithm for disambiguating words into *Roget's Thesaurus* categories

- One more knowledge based algorithm uses graphical representation of senses of words in context to disambiguate the term under consideration. (Navigli and Velardi)
- This is called the structural semantic interconnections (SSI) algorithm.
- It uses various sources of information like:
- WordNet
- Domain labels
- Annotated corpora to from structural specifications of concepts or semantic graphs
- The algorithm consists of two steps:

- An initialization
- An iterative step- the algorithm attempts to disambiguate all the words in the context iteratively.
- Its performance is very close to that of supervised algorithms and surpasses the best unsupervised algorithms.
- We now look at an example semantic graph for the two senses of the term bus.
- The first one is the vehicle sense

The second one is the connector sense

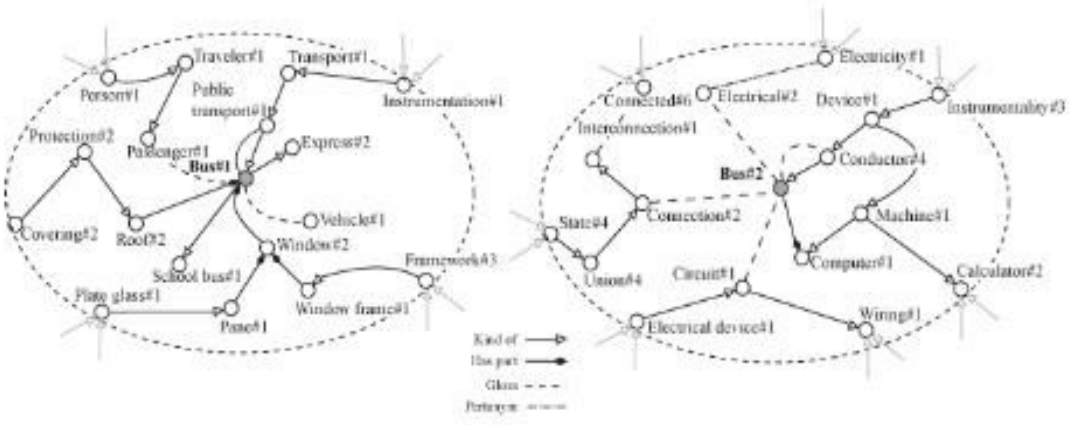


Figure 4–3. The graphs for sense 1 and 2 of the noun *bus* as generated by the SSI algorithm

ole concepts (or senses) of t.

- I (the semantic context) is the list of structural specifications of the concepts associated with each of the terms in T\{t} (except t). I=[S^{t1},S^{t2},…..,S^{tn}], that is the semantic interpretation of T.
- G is the grammar defining the various relations between the structural specifications.(semantic inter-connections among the graphs).
- Determine how well the structural specifications in I match that of S_1^t,S_2^t,……S_n^t using G.
- Select the best match S_i^t .
- The algorithm works as follows:
- A set of pending terms in the context P={t_i/S^{ti} =null} is maintained and I is used in each iteration to disambiguate terms in P.
- The procedure iterates and each iteration either:
- Disambiguates one term in P and removes it from the pending list
- Stops because no more terms can be disambiguated
- The output I is updated with the sense of t.
- Initially I contains structures for monosemous terms in T\{t} and any possible disambiguated synsets.

If this set is null then the algorithm makes an initial guess at what the most likely sense of the least ambiguous term in the contest is.

$$f_I(S,t) = \begin{cases} \rho(\{\varphi(S,S') | S' \in I\}), & \text{if } S \in Senses(t) \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

where $Senses(t)$ are the senses associated with the term t, and

$$\varphi(S,S') = \rho'(\{w(e_1 \cdot e_2 \cdots e_n) | S \xrightarrow{e_1} S_1 \xrightarrow{e_2} \dots \xrightarrow{e_{n-1}} S_{n-1} \xrightarrow{e_n} S'\}) \quad (4.2)$$

- The algorithm selects those terms t in P that show semantic interconnections with at least one sense of S of t and one or more senses in I.

- A function F_I(S,t) determines the likelihood of S being the correct interpretation of t and is defined as :
- A function ρ' of the weights w is the path connecting S and S'.
- A good choice for ρ and ρ' can be sum or average sum function.
- A CFG G = (E,N,SG,PG) encodes all the meaningful semantic patterns where:
- The other algorithms which are rule based are:

$$E = \{e_{kind-of}, e_{has-kind}, e_{part-of}, \dots\}$$

are the edge labels,

$$N = \{S_G, S_s, S_g, S_1, S_2, \dots, E_1, E_2, \dots\}$$

are nonterminal symbols that encode paths between the senses,

$$S_G$$

is the start symbol of the graph G, and

$$P_G = \{S_G \rightarrow S_s/S_g, S_s \rightarrow S_1/S_2/S_3, S_1 \rightarrow E_1S_1/E_1, E_1 \rightarrow e_{kind-of}/e_{part-of}, S_g \rightarrow e_{gloss}S_5/S_4/S_5, \dots\}$$

- Patwardhan, Benerjee and Pedersen compared several similarity measures based on WordNet.
- The emergence of Wikipedia has led to better applications in this area.
- Strube and Penzetto provided an algorithm called WikiRelate to estimate the distance between two concepts using the Wikipedia taxanomy.
- Navigli and Penzotto introduced an approach for creating a multilingual lexical knowledge base that establishes a mapping between Wikipedia and WordNet.

Supervised

- Supervised approach to disambiguation transfers all the complexity to the machine learning algorithm, but still requires hand annotation.
- The sense inventory for supervised approach must be predetermined and any changes needs a reannotation.
- The rules and knowledge can be incorporated in the form of features.
- A particular knowledge source and the classifier may have a few issues and the sense-tagged data could be noisy to varying degrees.
- We now look at a few algorithm which are useful for word sense disambiguation.

Supervised

- Brown was among the first few to use supervised learning and used the information in the form of parallel corpora.
- Yearowsky used a rich set of features and decision lists to tackle the word sense problem.
- A few researchers like Ng and Lee have come up with several variations including different levels of context and granularity.
- Now we look at the different **classifiers** and **features** that are relatively easy to obtain.

Supervised-Classifiers

- The most common and high performing classifiers are:
- Support Vector Machines
- Maximum Entropy Classifiers
- Since each lemma has a separate sense inventory a separate model is trained for each lemma and POS combination.

Supervised-Features

- Here we discuss a few commonly found subset of features that have been useful in supervised learning of word sense.
- The list provides a base that can be used to achieve nearly state-ofthe-art performance.
- The features are as follows:
- **Lexical context**- This feature comprises the words and lemmas of the words occurring in the entire paragraph or a smaller window of five words.
- **Parts of Speech**- This feature comprises the POS information for words in the window surrounding the word that is being sense tagged.

Supervised-Features

- **Bag of Words Context**- This feature comprises using an unordered set of words in the context window. A threshold is typically tuned to include the most informative words in the large context.
- **Local Collocations**
- They are an ordered sequence of phrases near the target word that provide semantic context for disambiguation.
- A very small window of about three tokens on each side of the target word in contiguous pairs of triplets are added as a list of features.
- For example for a target word w $C_{i,j}$ would be a collocation where I and j are the start and offset with respect to the word w.

- A positive sign indicates words to the right and negative sign indicates words to the left.

Supervised-Features-Example Local Collocations

- Let us look at an example:
- Ng and Lee used the following features:

- $C_{-1},-1$, $C_1,1$,$C_{-2},-2$,$C_2,2$,$C_{-2},-1$,$C_{-1},1$,$C_1,2$,$C_{-3},-1$,$C_{-2},1$,$C_{-1},2$ and $C_1,3$
- The example sentence "He bought a box of nails from the hardware store." where we try to disambiguate **nails.**

- The collection $C_{1,1}$ would be "from" and $C_{1,3}$ would be the string "from the hardware".

Stop words and punctuations are not removed before creating the collocations and for boundary conditions a null word is collocated

Syntactic Relations- If the parse of the sentence containing the target word is available then we can use syntactic features.

- A positive sign indicates words to the right and negative sign indicates words to the left.

Supervised-Features-Example Local Collocations

- Let us look at an example:
- Ng and Lee used the following features:

- $C_{-1},-1$, $C_1,1$,$C_{-2},-2$,$C_2,2$,$C_{-2},-1$,$C_{-1},1$,$C_1,2$,$C_{-3},-1$,$C_{-2},1$,$C_{-1},2$ and $C_1,3$
- The example sentence "He bought a box of nails from the hardware store." where we try to disambiguate **nails.**

- The collection $C_{1,1}$ would be "from" and $C_{1,3}$ would be the string "from the hardware".

Stop words and punctuations are not removed before creating the collocations and for boundary conditions a null word is collocated

Syntactic Relations- If the parse of the sentence containing the target word is available then we can use syntactic features.

Algorithm 4–2. Rules for selecting syntactic relations as features

---

1: **if** *w* is a *noun* **then**
2: *select* parent head word (*h*)
3: *select* part of speech of *h*
4: *select* voice of *h*
5: *select* position of *h* (left, right)
6: **else if** *w* is a *verb* **then**
7: *select* nearest word *l* to the left of *w* such that *w* is the parent head word of *l*
8: *select* nearest word *r* to the right of *w* such that *w* is the parent head word of *r*
9: *select* part of speech of *l*
10: *select* part of speech of *r*
11: *select* part of speech of *w*
12: *select* voice of *w*
13: **else if** *w* is a *adjective* **then**
14: *select* parent head word (*h*)
15: *select* part of speech of *h*
16: **end if**

• **Topic Features**- The broad topic or domain of the article that the word belongs to is also a good indicator of what sense of the word might be most frequent.

- Chen and Palmer proposed some additional features for disambiguation:
- **Voice of the Sentence**- This feature indicates whether the word occurs in passive, semipassive or active sentence.
- **Presence of subject/object**-
- This binary feature indicates whether the target word has a subject or object.
- We can also use the actual lexeme and semantic roles rather than the syntactic subject/object.
- **Sentential Complement**- This feature indicates whether the word has a sentential complement. (I know that he will do it.)
- **Prepositional Phrase Adjunct**- This feature indicates whether the target word has a prepositional phrase and if so selects the head of the noun phrase inside the prepositional phrase.
- **Named Entity**- This feature is the named entity of the proper nouns and certain types of common nouns.
- **WordNet**- WordNet synsets of the hypernyms of head nouns of the noun phrase arguments of verbs and prepositions.

Supervised-Features–Verb Sense Disambiguation

- **Path**- This feature is the path from the target verb to the verb's argument.
- **Subcategorization**- The subcategorization frame is essentially the string formed by joining the verb phrase type with that of its children.

UNSUPERVISED

- There is a problem in the progress of word sense disambiguation due to the lack of labeled training data to train a classifier.
- There are a few solutions to this problem:
- Devise a way to cluster instances of a word so that each cluster effectively constrains the examples of the word to a certain sense. This could be considered sense induction through clustering.
- Use some metrics to identify the proximity of a given instance with some sets of known senses of a word and select the closest to be the sense of that instance.
- Start with seeds of examples of certain senses then iteratively grow them to form clusters.
- We assume that there is already a predefined sense inventory for a word and that the unsupervised methods use very few hand-annotated examples and then attempt to classify unseen test instances into one of their predetermined sense categories.
- We first look at the category of algorithms that use some form of distance measure to identify senses.

Rada and others introduced a metric for computing the shortest distance between the two pairs of senses in WordNet

- This metric assumes that multiple co-occurring words exhibit senses that would minimize the distance in a semantic network of hierarchical relations. Ex: IS-A from WordNet.
- Resnik proposed a new measure of semantic similarity: **information content** in an IS-A taxonomy which produces much better results than the edge-counting measure.
- Agirre and Rigau refined this measure calling it **conceptual density** which not only depends on the number of separating edges but is also sensitive of the hierarchy and the density of its concepts.
- It is independent of the number of concepts being measured.
- Conceptual density is defined for each of the subhierarchies.

- The sense that falls in the subhierarchy with the highest conceptual density is chosen to be the correct sense.

- In the figure in the previous slide sense 2 is the one with the highest conceptual density and is therefore the chosen one.

Resnik observed that selectional constraints and word sense are related and identified a measure by which to compute the sense of a word on the basis of predicate argument statistics.

- This algorithm is primarily limited to the disambiguation of nouns that are arguments of verb predicates.

$$A_R(p,c) = \frac{1}{S_R(p)} P(c|p) \log \frac{P(c|p)}{P(c)}$$

- Let A_R be the selectional association of the predicate p to the concept c with respect to argument R. A_R is defined as:

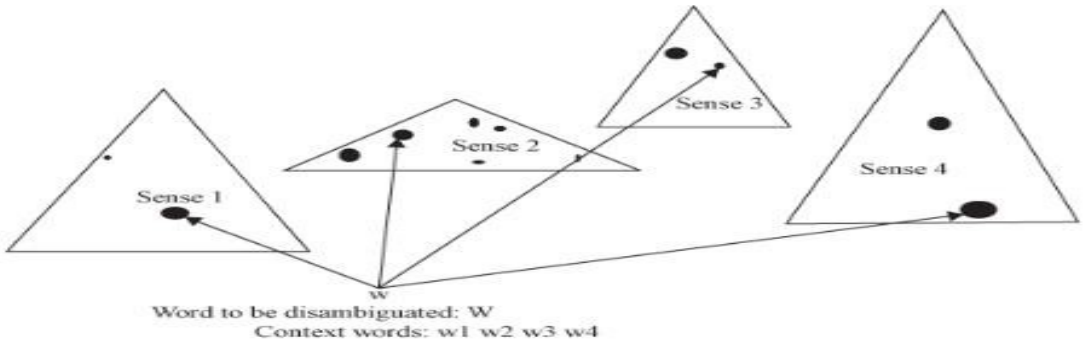


Figure 4–4. Conceptual density

$$CD(c,m) = \frac{\sum_{i=0}^{m-1} \text{hyponyms}^{i^{-0.20}}}{\text{descendants}_c} \qquad (4.3)$$

- If n is the noun that is in an argument relation R to predicated p, and {s1,s2,……,sk} are its possible senses, then, for i from 1 to k compute:

$$C_i = \{c | c \text{ is an ancestor of } s_i\}$$

$$a_i = \max_{c \in C_i} A_R(p, c)$$

- where ai is the score for sense si. The sense si which has the largest value of ai is sense for the word. Ties are broken by random choice.
- Leacock, Miller, and Chodorow provide another algorithm that makes use of corpus statistics and WordNet relations, and show that monosemous relatives can be exploited for disambiguating words.

Algorithms motivated by Cross-linguistic Evidence

Evidence

- There are a family of unsupervised algorithms based on crosslinguistic information or evidence.
- Brown and others were the first to make use of this information for purposes of sense disambiguation.
- They were interested in sense differences that required translating into other languages in addition to sense disambiguation.
- They provide a method to use the context information for a given word to identify its most likely translation in the target language.

Evidence

- Dagan and Itai used a bilingual lexicon paired with a monolingual corpus to acquire statistics on word senses automatically.
- They also proposed that syntactic relations along with word co-occurances statistics provide a good source to resolve lexical ambiguity.

- Diab performed experiments using machine translated English-to-Arabic translations to extract sense information for training a supervised classifier.
- Now let us look at a crosslinguistic algorithm.

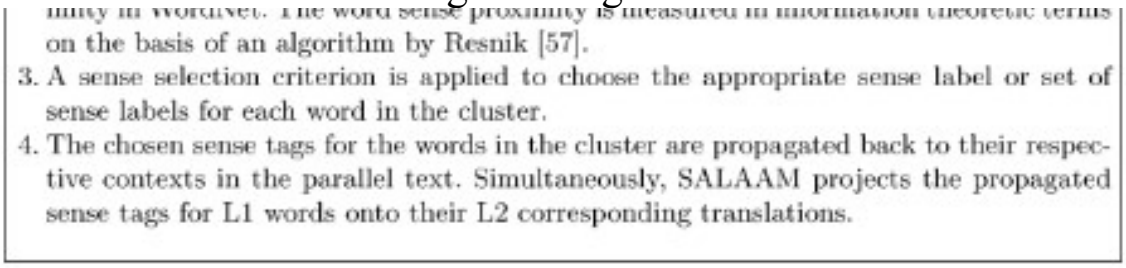


Figure 4–5. SALAAM algorithm for creating training using parallel English-to-Arabic machine translations

SEMI SUPERVISED

The next category of algorithms we look at are those that start from a small seed of examples and an iterative algorithm that identifies more training examples using a classifier.

- This additional automatically labeled data can be used to augment the training data of the classifier to provide better predictions for the next cycle.
- Yorowsky algorithm is such an algorithm which introduced semi-supervised methods to the word sense disambiguation problem.
- The algorithm is based on the assumption that two strong properties are exhibited by corpora:

- **One sense per collocation**: Syntactic relationship and the types of words occurring nearby a given word tend to provide a strong indication as to the sense of that word.
- **One sense per discourse**: In a given discourse all instances of the same lemma tend to invoke the same sense.

Based on the assumptions that these properties exist the Yarowsky algorithm iteratively disambiguates most of the words in a given discourse

A
u
s

Step 1. In a sufficiently large corpus, identify all the instances of a particular polysemous word that needs to be disambiguated, storing its context alongside.
Step 2. Identify a small set of instances that are strongly representative of one of the senses of the word. This can either be done in a completely unsupervised fashion by identifying collocations that give a strong indication of the sense usage for the word under consideration or by manually tagging a small portion of the data. In this example, we assume a polysemous word with only two senses, but this algorithm can be extended to n senses.
Step 3.

Step 3a. Train a supervised classifier on this set of examples.

Step 3b. Using these classifiers, classify the remaining instances of the word in the corpus and select those that are classified above a certain level of confidence.

Step 3c. Filter out the possible misclassifications using one sense per discourse constraint, and identify possible new collocations to be added to the list of seed collocations.

Step 3d. Repeat step 3 iteratively, thereby slowly shrinking the residual.

Step 4. Stop. At some point, a small, stable residual will remain.
Step 5. The trained classifier can now be used to classify new data, and that in turn can be used to annotate the original corpus with sense tags and probabilities.

Figure 4–7. The Yarowsky algorithm

- There are two presumptions here:
- One that the potential noise of wrong examples selected from a corpus during this process would be low enough so as not to effect learnability.
- Two that the overall discriminative ability of the model is superior to purely unsupervised methods or to situations in which not enough hand-annotated data is available to train a purely supervised system.
- Mihalcea and Moldovan describe an algorithm which is used to obtain examples from large corpora for particular senses from WordNet.
- Mihalcea proposed the following method using Wikipedia for automatic word sense disambiguation.
- Extract all the sense in Wikipedia in which the word under consideration is a link.

- There are two types of links: a simple link such as [[bar]] or a piped link such as [[musical_notation|bar]].
- Filter those links that point to a disambiguation page. This means that we need further information to disambiguate the word. If the word does not point to a disambiguation page the word itself can be the label.
- For all piped links the string before the pipe serves as the label.
- Collect all the labels associated with the word and then map them to possible WordNet senses.

  They might all map to the same sense essentially making the verb monosemous and not useful for this purpose

- The categories can be mapped to a significant number of WordNet categories there by providing sense-disamguated data for training.
- This algorithm provides a cheap way to extract sense information for many words that display the required properties.

Software

- Several software programs are available for word sense disambiguation.
- IMS (It makes Sense): This is a complete word sense disambiguation system
- WordNet Similarity-2.05: These WordNet similarity modules for Perl provide a quick way of computing various word similarity measures.
- WikiRelate: This is a word similarity measure based on categories in Wikipedia.

UNIT - IV

Predicate-Argument Structure, Meaning Representation Systems, Software

Predicate-Argument Structure

- Resources
- Systems
  - Software

Predicate-Argument Structure

- Shallow semantic parsing or semantic role labeling is the process of identifying the various arguments of predicates in a sentence.
- There has been a debate over what constitutes the set of arguments and what the granularity of such argument label should be for various predicates.

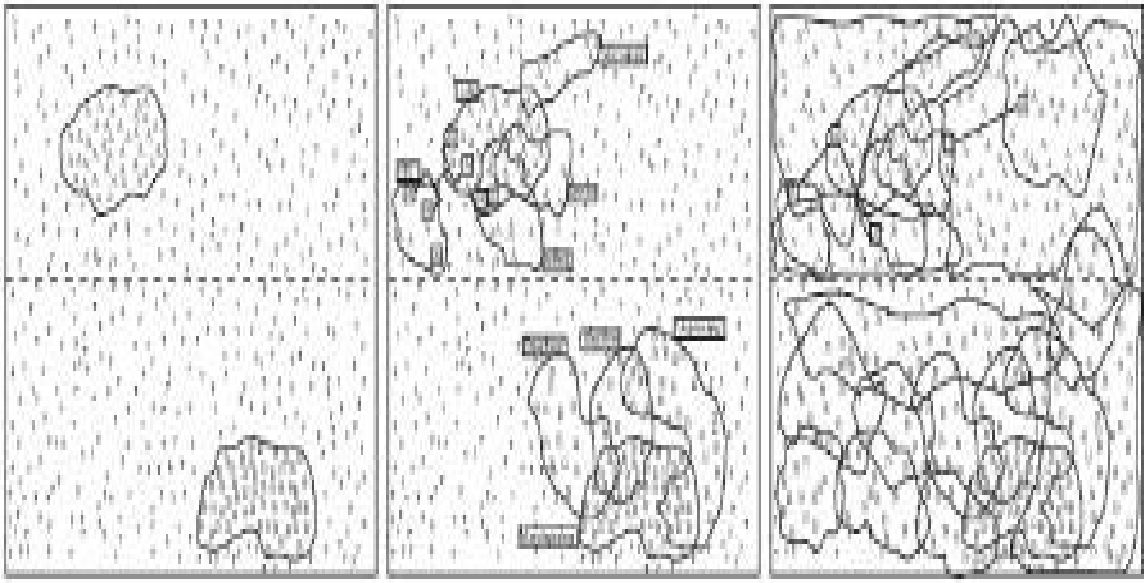


Figure 4–6. The three stages of the Yarowsky algorithm

Resources

- We have two important corpora that are semantically tagged. One is FrameNet and the other is PropBank.
- These resources have transformed from rule based approaches to more data-oriented approaches.
- These approaches focus on transforming linguistic insights into features.
- FrameNet is based on the theory of frame semantics where a given predicate invokes a semantic frame initiating some or all of the possible semantic roles belonging to that frame.

PropBank is based on Dowty's prototype theory and uses a more linguistically neutral view. Each predicate has a set of core arguments that are predicate dependent and all predicates share a set of noncore or adjunctive arguments

FRAME NET

- FrameNet contains frame-specific semantic annotation of a number of predicates in English.
- The process of FrameNet annotation consists of identifying specific semantic frames and creating a set of frame-specific roles called **frame elements**.
- A set of predicates that instantiate the semantic frame irrespective of their grammatical category are identified and a variety of sentences are labelled for those predicates.
- The labeling process identifies the following:
- The frame that an instance of the predicate lemma invokes
- The semantic arguments for that instance
- Tagging them with one of the predetermined set of frame elements for that frame.
- The combination of the predicate lemma and the frame that its instance invokes is called a lexical unit (LU).
- Each sense of a polysemous word tends to be associated with a unique frame.
- The verb “break” can mean fail to observe (a law, regulation, or agreement) and can belong to a COMPLIANCE frame along with other word meanings such as violation, obey, flout.
- It can also mean cause to suddenly separate into pieces in a destructive manner and can belong to a CAUSE_TO_FRAGMENT frame along with other meanings such as fracture, fragment, smash.

- Here the frame **Awareness** is instantiated by the verb predicate **believe** and the noun predicate **comprehension**.

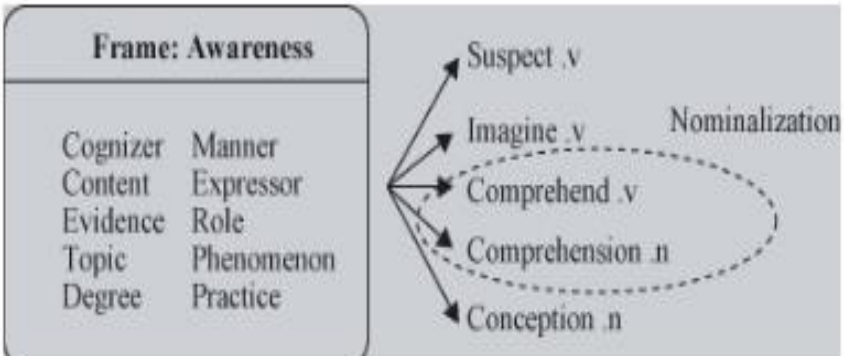


Figure 4–9. FrameNet example

- **1.** [$_{Cognizer}$ We] [$_{Predicate:verb}$ *believe*] [$_{Content}$ it is a fair and generous price]
- **2.** No doubts existed as to [$_{Cognizer}$ our] [$_{Predicate:noun}$ *comprehension*] [$_{Content}$ of it]

units.

- Example:

The frame element BODY_PART in frame CURE has the same meaning as the same element in the frame GESTURE or WEARING

PROPBANK

- PropBank includes annotations of arguments of verb predicates.
- PropBank restricts the argument boundaries to that of a syntactic constituent as defined in the Penn Treebank.
- The arguments are tagged either:
- **Core arguments** with labels of type ARGN where N takes values from 0 to 5.
- **Adjunctive arguments** with labels of the type ARGM-X where X can take values such as TMP for temporal LOC for locative etc.

Tag	Description	Examples
ARGM-LOC	Locative	*the museum, in Westborough, Mass.*
ARGM-TMP	Temporal	*now, by next summer*
ARGM-MNR	Manner	*heavily, clearly, at a rapid rate*
ARGM-DIR	Direction	*to market, to Bangkok*
ARGM-CAU	Cause	*In response to the ruling*
ARGM-DIS	Discourse	*for example, in part, Similarly*
ARGM-EXT	Extent	*at $38.375, 50 points*
ARGM-PRP	Purpose	*to pay for the plant*
ARGM-NEG	Negation	*not, n't*
ARGM-MOD	Modal	*can, might, should, will*
ARGM-REC	Reciprocals	*each other*
ARGM-PRD	Secondary Predication	*to become a teacher*
ARGM	Bare ARGM	*with a police escort*
ARGM-ADV	Adverbials	*(none of the above)*

- Adjunctive arguments share the same meaning across all predicates.

The meaning of core arguments has to be interpreted in connection with a predicate.

Let us look at an example from PropBank corpus along with its syntax tree

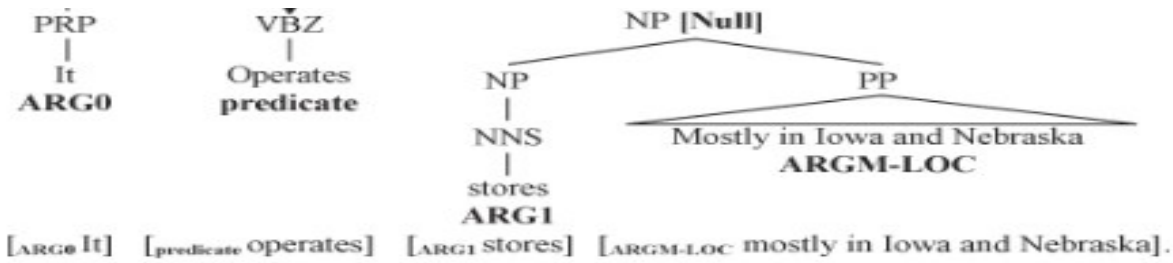


Figure 4–10. Syntax tree for a sentence illustrating the PropBank tags

- Most Treebank-style trees have **trace nodes** that refer to another node in the tree but have no words associated with them.
- These can also be marked as arguments.
- Since traces are not reproduced by a usual syntactic parser the community has disregarded them from most standard experiments.
- There are a few disagreements between Treebank and PropBank. In such cases the a sequence of nodes in the tree are annotated as the argument and called as **discontinuous arguments**.

FrameNet Vs Propbank

- An important distinction between FrameNet and Propbank is as follows:
- In FrameNet we have lexical units which are words paired with their meanings or the frames that they invoke.
- In Propbank each lemma has a list of different framesets that represent all the senses for which there is a different argument structure.
- Other resources have been developed to aid further research in predicate-argument recognition.
- NomBank was inspired by PropBank.

OTHER RESOURCSES

- In the process of identifying and tagging the arguments of nouns, the NOMLEX (NOMinalization LEXicon) dictionary was expanded to cover about 6,000 entries.

work), and for *author.01* (sense: to write or construct) in the PropBank corpus

Predicate	Argument	Description
operate.01		
	ARG0	Agent, operator
	ARG1	Thing operated
	ARG2	Explicit patient (thing operated on)
	ARG3	Explicit argument
	ARG4	Explicit instrument
author.01		
	ARG0	Author, agent
	ARG1	Text authored

to other languages.

- Since the nature of semantics is lingua independent frames can be reused to annotate data in other languages.
- The SALSA project was the first to put this into practice.
- Since FrameNet tags both literal and metaphorical interpretation SALSA project remained close to lexical meaning.
- There are FrameNets in other languages like Japanese, Spanish and Swedish.
- PropBank has inspired creation of similar resources in Chinese, Arabic, Korean, Spanish, Catalan, and Hindi.
- Every new PropBank requires the creation of new set of frame files unlike FrameNet.
- FrameNet and PropBank are not the only styles used in practice.
- Prague Dependency TreeBank tags the predicate argument structure in its tactogrammatic layer on top of dependency structure.
- It also makes a distinction same as core and adjunctive arguments called **inner participants** and **free modifications**.
- The NAIST text corpus is strongly influenced by the traditions in Japanese linguistics.

SYSTEMS

- Syntactic Representations
- Classification Paradigms
- Overcoming the Independence Assumptions
- Feature Performance
- Feature Salience
- Feature Selection
- Size of Training Data
- Overcoming Parsing Errors
- Noun Arguments

Multilingual Issues

- Robustness across Genre
- Very little research has gone into learning predicate argument structures from unannotated corpora.
- The reason is predicate-argument structure is closer to the actual applications and has been very close to the area of information extraction.
- Early systems in the area of predicate-argument structures were based on heuristics on syntax tree which were rule based.
- A few of the early systems were:
- The Absity parser PUNDIT understanding system were among the early rule based systems.
- One hybrid method for thematic role tagging using WordNet as a resource was introduced.
- Other notable applications are:
- Corpus based studies by Manning, Briscoe, and Carroll which seek to derive the subcategorization information from large corpora
- Pustejovsky which tries to acquire lexical semantic knowledge from corpora
- A major step in semantic role labelling research happened after the introduction of FrameNet and PropBank.
- One problem with these corpora is significant work goes into creating the frames, that is, in classifying verbs into framesets in preparation for manual annotation.
- Providing coverage for all possible verbs in one or more languages require significant manual effort.
- Green, Dorr and Resnik propose a way to learn the frame structures automatically but the result is not accurate enough to replace the manual frame creation.
- Swier and Stevenson represent one of the more recent approaches to handling this problem in an unsupervised fashion.

Let us now look at a few applications after the advent of these corpora

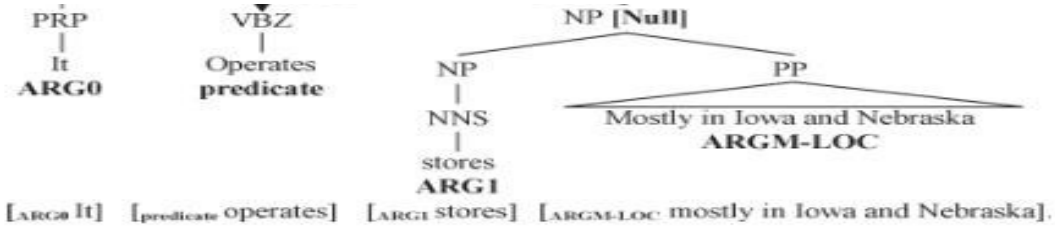


Figure 4–10. Syntax tree for a sentence illustrating the PropBank tags

- In the example sentence in the previous slide, for the predicate operates, the word “It” fills with the role ARG0, the word

"stores" fills the ARG1, and the sequence of words "mostly in Iowa and Nebraska" fills the role ARGM-LOC.

- An ARGN for one predicate need not have similar semantics compared to another predicate.
- FrameNet was the first project that used hand-tagged arguments of predicates in data.
- Gildea and Jurafsky formulated semantic role labeling as a supervised classification problem that assumes the arguments of the predicate.
- The predicate itself can be mapped to a node in the syntax tree of that sentence.

They introduced three tasks which can be used to evaluate the system

- **Argument Identification**: This is the task of identifying all and only the parse constituents that represent valid semantic arguments of a predicate.
- **Argument Classification**: Given constituents known to represent arguments of a predicate, assign the appropriate argument labels to them.
- **Argument identification and classification**: This task is a combination of the previous two tasks where the constituents that represent arguments of a predicate are identified and the appropriate argument label is assigned to them.
- After parsing each node in the parse tree can be classified as:
- One that represents a semantic argument (non-null node)
- One that does not represent any semantic argument (null node)
- The non-null node can further be classified into the set of argument labels.

- In the previous tree the noun phrase that encompasses “mostly in Iowa and Nebraska” is a null node because it does not correspond to a semantic argument.

The node NP that encompasses “stores” is a non-null node because it does correspond to a semantic argument: ARG1

- he pseudo code for a generic semantic role labeling(SRL) algorithm is as follows:

Syntactic Representation

- Phrase Structure Grammar

Algorithm 4–3. The semantic role labeling (SRL) algorithm

Procedure: SRL(sentence) **returns** best *semantic role labeling*

Input: *syntactic parse* of the *sentence*

1: *generate* a full syntactic parse of the *sentence*
2: *identify* all the *predicates*
3: **for all** *predicate* ∈ *sentence* **do**
    4: *extract* a set of features for each node in the tree relative to the *predicate*
    5: *classify* each feature vector using the *model* created in training
    6: *select* the class of highest scoring classifier
    7: **return** best *semantic role labeling*
8: **end for**

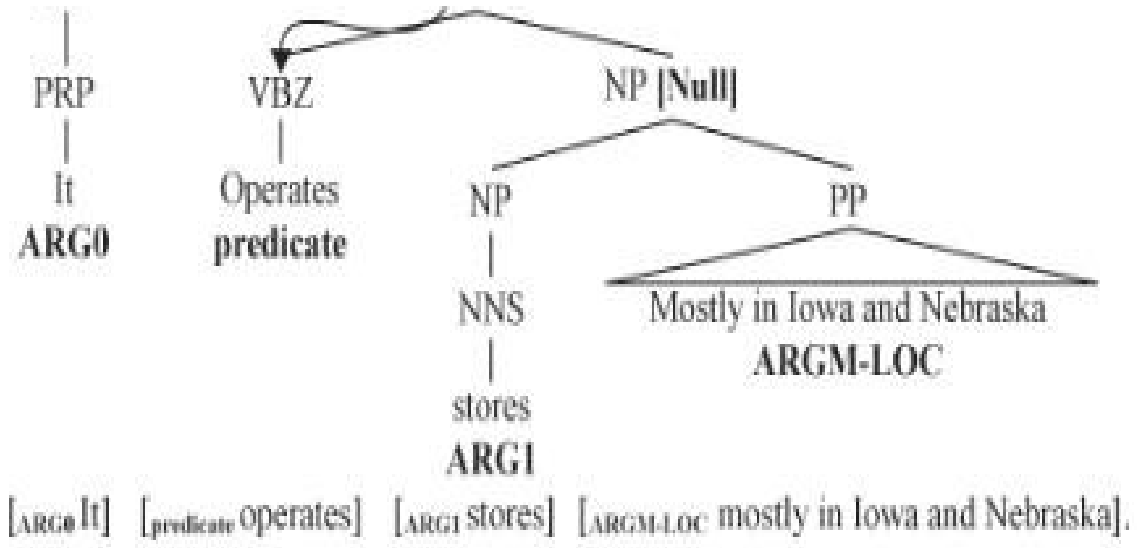
- Combinatory Categorial Grammar
- Tree Adjoining Grammar

Syntactic Representations

- PropBank was created as a layer of annotation on top of Penn TreeBank style phrase structure trees.
- Gidea and Jurafsky added argument labels to parses obtained from a parser trained on Penn TreeBank .
- Researchers have also used other types of sentence representations to tackle the semantic role labeling problem.
- We now look at a few of these sentence representations and the features that were used to tag text with PropBank arguments.

Phrase Structure Grammar

- FrameNet marks word spans in sentences to represent arguments whereas PropBank tags nodes in a treebank tree with arguments.
- Since the phrase structure representation is amenable to tagging Gildea and Jurafsky introduced the following features:
- **Path**: This feature is the syntactic path through the parse tree from the parse constituent to the predicate being classified.
- For example:
- In the figure in the next slide the path from ARG0 “It” to the predicate “operates” is represented by the string NP↑ □ ↓ □□ ↓ □□□.



- **Predicate**: The identity of the predicate lemma is used as a feature.
- **Phrase Type**: This feature is the syntactic category (NP, PP, S, etc.) of the constituent to be labeled.
- **Position**: This feature is a binary feature identifying whether the phrase is before or after the predicate.
- **Voice**: This feature indicates whether the predicate is realized as an active or passive construction. A set of hand written expressions on the syntax tree are used to identify the passive-voiced predicates.
- **Head Word**: This feature is the syntactic head of the phrase. It is calculated using a head word table.
- **Subcategorization**: This feature is the phrase structure rule expanding the predicate's parent node in the parse tree.
- For example:
- In the figure in the previous slide the subcategorization for the predicate "operates" VP ☾ VBZ-NP.

- **Verb Clustering**:
- This predicate is one of the most salient features in predicting the argument class.
- Gildea and Jurafsky used a distance function for clustering that is based on the intuition that verbs with similar semantics will tend to have similar direct objects.
- For example:
- Verbs such as eat, devour and savor will occur with direct objects describing food.
- The clustering algorithm uses a database of verb-direct object relations.
- The verbs were clustered into 64 classes using the probabilistic co-occurrence model.
  Surdeanu suggested the following features

Content Word: Since in some cases head words are not very informative a different set of rules were used to identify a so-called **content** word instead of using the head-word finding rules. The rules that they used are

```
H1: if phrase type is PP then select the rightmost child
    Example: phrase = "in Texas," content word = "Texas"
H2: if phrase type is SBAR then select the leftmost sentence (S*) clause
    Example: phrase = "that occurred yesterday," content word = "occurred"
H3: if phrase type is VP then
        if there is a VP child then
            select the leftmost VP child
        else
          select the head word
    Example: phrase = "had placed," content word = "placed"
H4: if phrase type is ADVP then select the rightmost child, not IN or TO
    Example: phrase = "more than," content word = "more"
H5: if phrase type is ADJP then select the rightmost adjective, verb, noun, or ADJP
    Example: phrase = "61 years old," content word = "61"
H6: for all other phrase types select the head word
    Example: phrase = "red house," content word = "red"
```

Figure 4–11. List of content word heuristics

- **POS of Head Word and Content Word**: Adding the POS of the head word and the content word of a constituent as a feature to help generalize in the task of argument identification and gives a performance boost to their decision tree-based systems.
- **Named Entity of the Content Word**: Certain roles such as ARGM-TMP, ARGM-LOC tent to contain time or place named entities. This information was added as a set of binary valued features.
- **Boolean Named Entity Flags**: Named entity information was also added as a feature. They created indicator functions for each of the seven named entity types: PERSON, PLACE, TIME, DATE, MONEY, PERCENT, ORGANIZATION.
- **Phrasal Verb Collocations**: This feature comprises frequency statistics related to the verb and the immediately following preposition.
- Fleischman, Kwon, and Hovy added the following features to their system:
- **Logical function**: This is a feature that takes three values external argument, object argument, and other argument and is computed using some heuristic on the syntax tree.

- **Order of Frame Elements**: This feature represents the position of a frame element relative to other frame elements in a sentence.
    - **Syntactic Pattern**: This feature is also generated using heuristics on the phrase type and the logical function of the constituent.
    - **Previous Role**: This is a set of features indicating the nth previous role that had been observed/assigned by the system for the current predicate.
    - Pradhan suggested using the following additional features:
    - **Named Entities in Constituents**:
    - Named entities such as location and time are important for the adjunctive arguments ARGM-LOC and ARGM-TMP.
    - Entity tags are also helpful in cases where head words are not common.
    - Each of these features is true if its representative type of named entity is contained in the constituent.
    - **Verb Sense Information**:
    - The arguments that a predicate can take depend on the sense of the predicate.
    - Each predicate tagged in the PropBank corpus is assigned a separate set of arguments depending on the sense in which it is used.
    - This is also known as the **frameset ID.**
    - The table below illustrates the argument sets for a word. Depending on the sense of the predicate "talk" either ARG-1 or ARG-2 can identify the hearer.

Tag	Description	Tag	Description
ARG0	Talker	ARG0	Talker
ARG1	Subject	ARG1	Talked to
ARG2	Hearer	ARG2	Secondary action

- sense of a predicate as a distinct predicate which helps performance.
- This disambiguation of PropBank framesets can be performed at a very high accuracy.
- **Noun head of prepositional phrases**:
- For instance “in the city” and “in few minutes” both share the same head word “in”.
- The former is ARG-LOC whereas the latter is ARG-TMP.
- the prepositional phrase.
- The prepositional information is retained by appending it to the phrase type. (Ex PP-IN)
- **First and Last Word/POS in Constituent**: Some arguments tent to contain discriminative first and last words, so these were used along with their POS as four new features.
- **Ordinal Constituent Position**: This feature avoids false positives where constituents far away from the predicate are spuriously identified as arguments.
- **Constituent Tree Distance**: This is finer way of specifying the already present position feature, where the distance of the constituent from the predicate is measured in terms of the number of nodes that need to be traversed through the syntax tree to go from one to the other.
- **Constituent relative features**: These are features representing the constituent type, head word and head word POS of the parent, and left and right siblings of the constituent in focus. This is added for robustness and to improve generalization.
- **Temporal cue words**: Several temporal cue words are not captured by the named entity tagger and were therefore added as binary features indicating their presence. (Temporal words specify order)
- **Dynamic class context**: In the task of argument classification, there are dynamic features that represent the hypothesis of at most the previous two non-null nodes belonging to the same tree as the node being classified.
- **Path generalizations**:

- The path is one of the most salient feature for the argument identification.
- This is also the most data-sparse feature.
- To overcome this problem the path was generalized in several different ways
- **Clause-based path variations**: Position of the clause node (S,SBAR) seems to be an important feature in argument identification. Experiments were done with four clause-based path feature variations.
- Replacing all the nodes in the path other than clause nodes with an (*).

Retaining only the clause nodes in the path

- Adding a binary feature that indicates whether the constituent is in the same clause as the predicate.
- Collapsing the nodes between S nodes.
- **Path n-grams**: This feature decomposes a path into a series of trigrams. For example the path NP↑ ↑↑↑↑↑VP↓ VBD becomes NP↑ ↑↑, ↑↑↑↑ and so on. Shorter paths can be padded with nulls.
- **Single-Character phrase tags**: Each phrase category is clustered to a category defined by the first character of the phrase label.
- **Path compression**: Compressing sequences of identical labels into one following the intuition that successive embedding of the same phrase in the tree might not add additional information.
- **Directionless path**: Removing the direction in the path, thus making insignificant the point at which it changes direction in the tree.

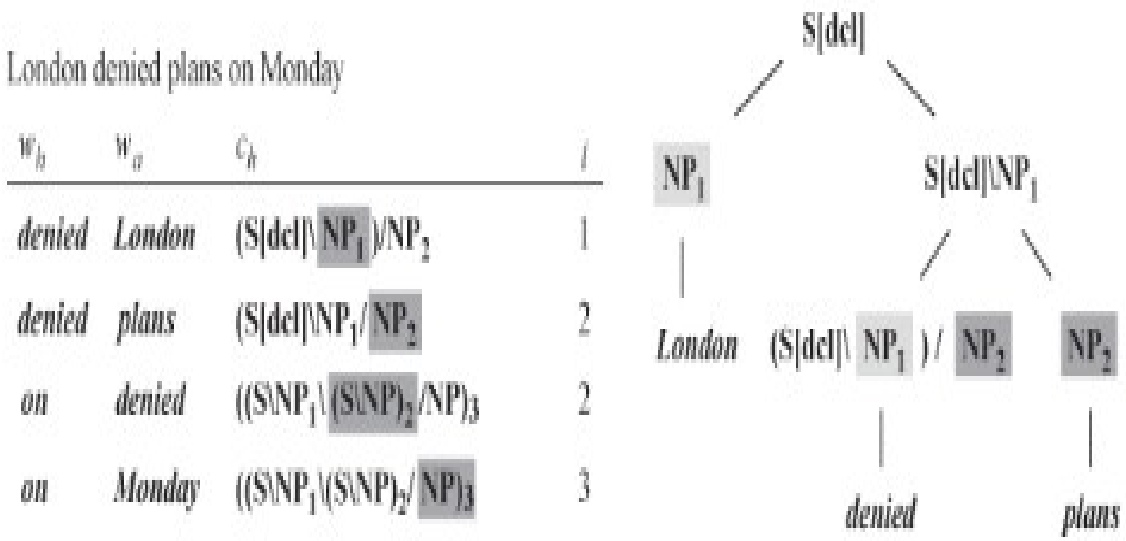
- **Partial path**: Using only that part of the path from the constituent to the lowest common ancestor of the predicate and the constituent.
- **Predicate context**: This feature captures predicate sense variations. Two words before and two words after were added as features. The POS of the words were also added as features.
- **Punctuations**: For some adjunctive arguments, punctuation plays an important role. This set of features captures whether punctuation appears immediately before and after the constituent.
- **Feature context**:
- Features of constituents that are parent or siblings of the constituent being classified were found useful.
- There is a complex interaction between the types and number of arguments that a constituent can have.

This feature uses all the other vector values of the constituents that are likely to be non-null as an added context

Combinatory Categorial Grammar

- Though the path feature is important for argument identification task, it is one of the sparse features and difficult to train or generalize.
- Dependency parsers generate shorter paths from the predicate to dependent words in the sentence and can be robust complement to the paths extracted from the PSG parse tree.
- Using features extracted from a CCG representation improves semantic role labeling performance on core arguments.
- CCG trees are binary trees and the constituents have poor alignment with the semantic arguments of a predicate.

-



Let us look at an example of the CCG parse of the sentence “London denied plans on Monday”

- Gildea and Hockenmaier introduced three features:
- **Phrase type**: This is the category of the maximal projection between the two words, the predicate and the dependent word.
- **Categorial path**: This is a feature formed by concatenating the following three values:
- Category to which the dependent word belongs
- The direction of dependence
- The slot in the category filled by the dependent word
- The path between denied and plans in the previous figure would be:
- (S[dcl]\NP)/NP←.

Tree Path: This is the categorical analogue of the path feature in the Charniak parse based system, which traces the path from the dependent word to the predicate through the binary CCGtree.

Tree-Adjoining Grammar (TAG)

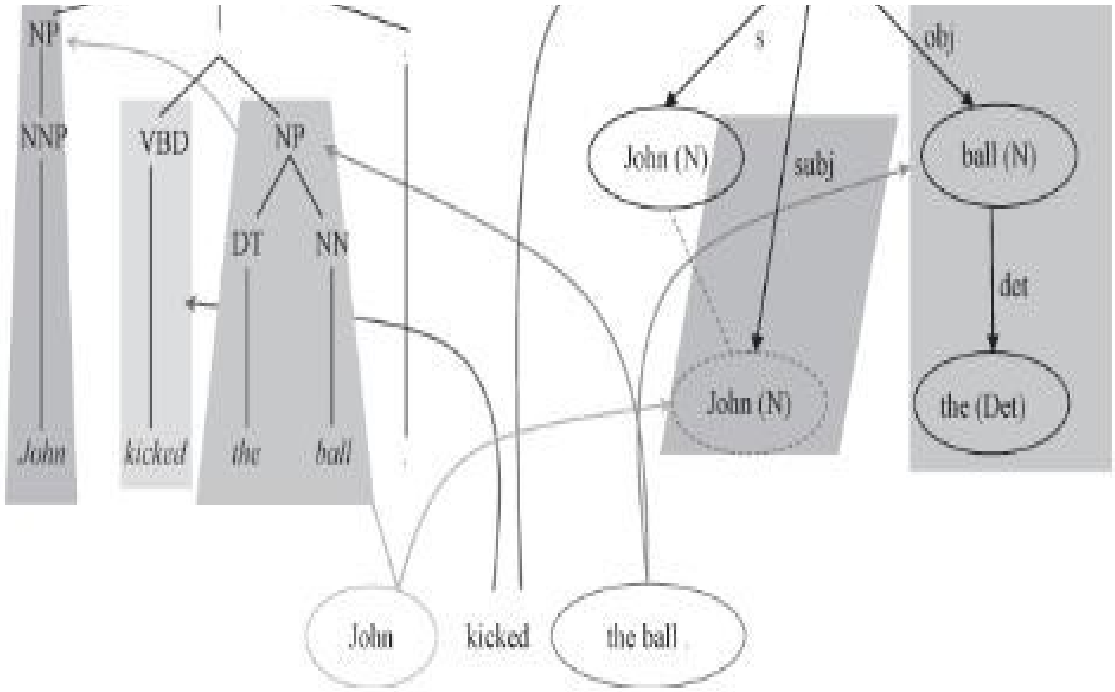
- TAG has an ability to address long-distance dependencies in text.
- The additional features introduced by Chen and Rambow are:
- **Supertag path**: This feature is the same as the path feature seen earlier except that in this case it is derived from a TAG rather than from a PSG.
- **Supertag**: This feature is the tree frame corresponding to the predicate or the argument.
- **Surface syntactic role**: This feature is the surface syntactic role of the argument.
- **Surface subcategorization**: This feature is the subcategorization frame.
- **Deep syntactic role**: This feature is the deep syntactic role of an argument, whose values include subject and direct object.
- **Deep subcategorization**: This is the deep syntactic subcategorization frame.

- **Semantic subcategorization**: Gildea and Palmer also used a semantic subcategorization frame where, in addition to the syntactic categories, the feature includes semantic role information.
- A few researchers tried to use a tree kernel that identified and selected subtree patterns from a large number of automatically generated patterns to capture the tree context.
- The performance of this automated process is not as good as handcrafted features.

Dependency Trees

- One issue so far is that the performance of a system depends on the exact span of the arguments annotated according to the constituents in the Penn Treebank.
- PropBank and most syntactic parsers are developed in the Penn Treebank corpus.
- They will match the PropBank labeling better than the other representations.
- Algorithms which depend on the relation of the argument head word to the predicate give much higher performance with an F-score of about 85.
- Hacioglu formulated the problem of semantic role labeling on a dependency tree by converting the Penn Treebank trees to a dependency representation.
- They used a script by Hwa, Lopez, and Diab and created a dependency structure labeled with PropBank arguments.
- The performance on this system seemed to be about 5 F-score points better than the one trained on the phrase structure trees.
- Parsers trained on Penn Treebank seem to degrade in performance when evaluated on sources other than WSJ.
- Minipar is a rule-based dependency parser that outputs dependencies between a word called head and another called modifier.

- The dependency relationships form a dependency tree.
- The set of words under each node in Minipar's dependency tree form a contiguous segment in the original sentence and correspond to the constituent in a constituent tree.



- The figure in the previous slide shows how the arguments of the predicate "kick" map to the nodes in a phrase structure grammar tree as well as the nodes in a Minipar parse tree.
- The nodes that represent head words of constituents are the targets of classification.

They used the features in the following slide

POS path	The path from the predicate to the head word through the dependency tree connecting the part of speech of each node in the tree.
Dependency path	Each word that is connected to the head word has a dependency relationship to the word. These are represented as labels on the arc between the words. This feature comprises the dependencies along the path that connects two words.
Voice	Voice of the predicate.
Position	Whether the node is before or after the predicate.

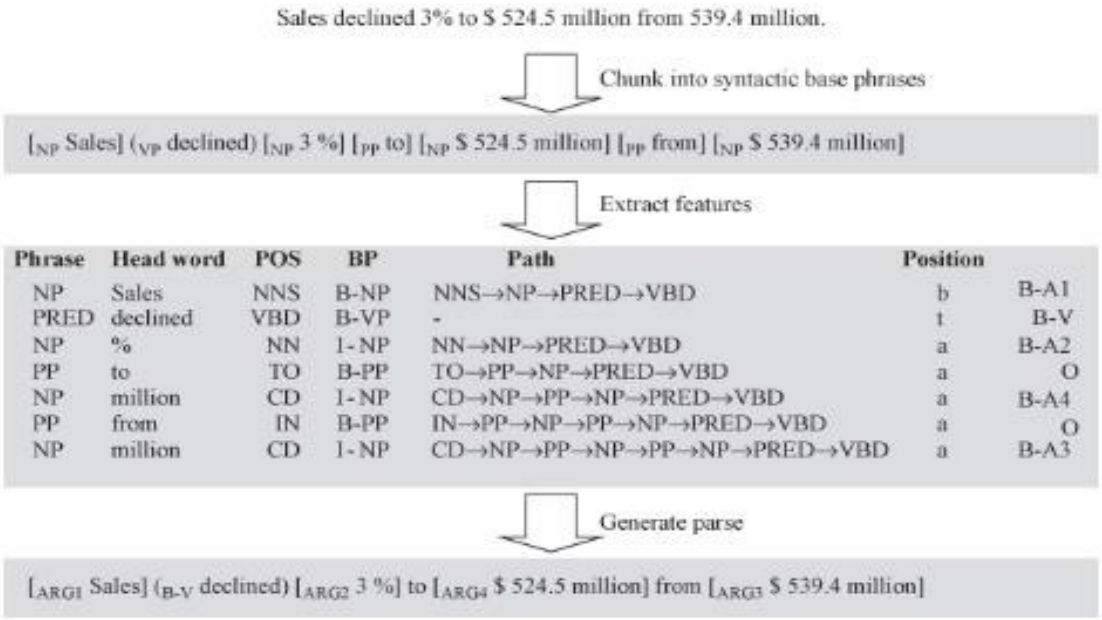
- An important question is how much does the full syntactic representation help the task of semantic role labeling?
- How important is it to create a full syntactic tree before classifying the arguments of a predicate?
- A chunk representation can be faster and more robust to phrase reordering as in the case of speech data.
- It was concluded that syntactic parsing helps fill a big gap using chunk based approach by Gildea and Palmer.

Chunking based systems classify each base phrase as the B(eginning) of a semantic role, I(nside) a semantic role, or O(utside) any semantic

- This is referred to as an IOB representation.
- This system uses SVM classifier to first chunk input text into flat chunks or base phrases, each labeled with a syntactic tag.

- A second SVM is trained to assign semantic labels to the chunks.

Figure in the next slide shows a schematic of the chunking process



The following table lists the features used by the semantic chunker

Words	Words in the chunk.
Predicate lemma	The predicate lemma.
POS tags	Part of speech of the words in the chunk.
BP positions	The position of a token in a base phrase (BP) using the IOB2 representation (e.g., B-NP, I-NP, O).
Clause tags	The tags that mark token positions in a sentence with respect to clauses.
Named entities	The IOB tags of named entities.
Token position	The position of the phrase with respect to the predicate. It has three values: "before," "after," and "-" (for the predicate).
Path	Defines a flat path between the token and the predicate.
Clause bracket patterns	
Clause position	A binary feature that identifies whether the token is inside or outside the clause containing the predicate.
Headword suffixes	Suffixes of head words of length 2, 3, and 4.
Distance	Distance of the token from the predicate as a number of base phrases and the distance as the number of VP chunks.
Length	The number of words in a token.
Predicate POS tag	The part of speech category of the predicate.
Predicate frequency	Frequent or rare using a threshold of 3.
Predicate BP context	The chain of BPs centered at the predicate within a window of size $-2/+2$.
Predicate POS context	POS tags of words immediately preceding and following the predicate.
Predicate-argument frames	Left and right core argument patterns around the predicate.
Number of predicates	This is the number of predicates in the sentence.

Classification Paradigms

- Here we focus on the ways in which machine learning has been brought to bear on the problem of semantic role labeling.
- The simplest approaches are those that view semantic role labeling as a pure classification problem.
- Here each argument of a predicate may be classified independent of others.

- A few researchers have adopted the same basic paradigm but added a simple postprocessor that removes implausible analysis, such as when two arguments overlap.
- A few more complicated approaches augment the post processing step to use argument specific language models or frame element group statistics.
- There are more sophisticated approaches to perform joint decoding of all the arguments, trying to capture the arguments interdependence.
- These sophisticated approaches have yielded only slight gains because the performance of a pure classifier followed by a simple postprocessor is already quite high.
- Here we concentrate on a current high-performance approach that is very effective.
- Let us look at the process of the SRL algorithm designed by Gildea and Jurafsky. It involves two steps:
- In the first step:
- It calculates the maximum likelihood probabilities that the constituent is an argument based on two features:
- P(argument/Path, Predicate)
- P(argument/Head, Predicate)
- It interpolates them to generate the probability that the constituent under consideration represents an argument.
- In the second step:
- It assigns each constituent that has a nonzero probability of being an argument a normalized probability calculated by interpolating distributions conditioned on various sets of features.
- It then selects the most probable argument sequence

- Surdeanu used a decision tree classifier C5 on the same features as Gildea and Jurafsky.
- Chen and Rambow used decision tree classifier C4.5 algorithm.
- Fleischman and Hovy report results on the FrameNet corpus using a maximum entropy framework.
- Pradhan used SVM for the same and got even better performance on the PropBank corpus.
- The difference in the result between SVM and entropy classifier is very small.
- The following table compares a few argument classification algorithms using same features:

same features but different classifiers

Classifier	Accuracy (%)
SVM (Pradhan et al.) [120]	88
Decision Tree (Surdeanu et al.) [118]	79
Gildea and Palmer [131]	77

Table 4–6. Distributions used for semantic argument classification, calculated from the features extracted from a Charniak parse

Distributions
P(argument|Predicate)
P(argument|Phrase Type, Predicate)
P(argument|Phrase Type, Position, Voice)
P(argument|Phrase Type, Position, Voice, Predicate)
P(argument|Phrase Type, Path, Predicate)
P(argument|Phrase Type, Path, Predicate, Subcategorization)
P(argument|Head Word)
P(argument|Head Word, Predicate)
P(argument|Head Word, Phrase Type, Predicate)

- SVMs performs well on text classification tasks where data are represented in a high dimensional space using sparse feature vectors.
- Pradhan formulated the semantic role labeling problem as a multiclass classification problem using SVMs.
- SVMs are inherently binary classifiers but multiclass problems can be reduced to a number of binary-class problems using either the pairwise approach or the one versus all (OVA) approach.
- For an N class problem in the pairwise approach, a binary classifier is trained for each pair of the possible N(N-1)/2 class pairs.
- In the OVA approach, N binary classifiers are trained to discriminate each class from a metaclass created by combining the rest of the classes.
- The SVM system can be viewed as comprising two stages:
- The training stage
- The testing stage
- The training stage is divided into two stages:
- In the first stage:
- Filter out the nodes that have a very high probability of being null
- A binary classifier is trained on the entire dataset
- Fit a sigmoid function to the raw scores to convert the scores to probabilities.
- The respective scores for all the examples are converted to probabilities using the sigmoid function
- Nodes that are most likely null (probability >.90) are pruned from training set.
- In the second stage the remaining training data are used to train OVA classifiers for all the classed along with a null class.
- With this strategy only one classifier has to be trained on all of the data.
- The remaining classifiers are trained on the nodes passed by the filter.
- In the testing stage all the nodes are classified directly as null or one of the arguments using the classifier trained in step 2.

- A variation in this strategy would be to filter all the examples that are null in the first pruning stage instead of just pruning out the highprobability ones.
- On gold-standard Treebank parsers the performance of such a system on the combined task of argument identification and classification is in the low 90s.
- On automatically generated parsers the performance tends to be in the high 70s.

Overcoming the Independence Assumption

- Various post-processing stages have been proposed to overcome the limitations of treating semantic role labeling as a series of independent argument classification steps. Some of these strategies are:
- Disallowing Overlaps
- Argument Sequence Information

Disallowing Overlaps

- Since each constituent is classified independently it is possible that two constituents that overlap get assigned the same argument type.
- Since we are dealing with parse tree nodes overlapping in words always have an ancestor-descendent relationship.
- The overlaps are restricted to subsumptions only.
- Since overlapping arguments are not allowed in PropBank, one way to deal with this issues is to retain one of them.
- We retain the one for which the SVM has the highest confidence based on the classification probabilities.
- The others are labeled Null.

Example 4–1: But [$_{ARG0}$ nobody] [$_{predicate}$ knows] [$_{ARG1}$ at what level [$_{ARG1}$ the futures and stocks will open today]]

The probabilities obtained by applying the sigmoid function to the raw SVM score are used as the measure of confidence

Argument Sequence Information

- One more way of overcoming the independence assumption is the use of the fact that a predicate is likely to instantiate a certain set of argument types to improve the performance of the statistical argument tagger.
- A better approach involves imposing additional constraints in which argument ordering information is retained and the predicate is considered as an argument and is part of the sequence.
- To achieve this we train a trigram language model on the argument sequence .
- We first convert the raw SVM scores to probabilities.

Argument Sequence Information

- After that for each sentence an argument lattice is generated using the n-best hypotheses for each node in the syntax tree.
- A Viterbi search is then performed through the lattice using the probabilities assigned by the sigmoid function.
- The probabilities are assigned as the observed probabilities along with the language model probabilities to find the maximum likelihood path through the lattice such that each node is assigned a value belonging to the PropBank arguments or null.
- The search is constrained in such a way that no two non-null nodes overlap.
- To simplify the search we must allow only null assignments to nodes having a null likelihood above a threshold.

- It was found that there was an improvement in the core argument accuracy whereas the accuracy of the adjunctive arguments slightly deteriorated.

FEATURES	ARGUMENT CLASSIFICATION	ARGUMENT IDENTIFICATION		
	A	P	R	F_1
Baseline [120]	87.9	93.7	88.9	91.3
+ Named entities	**88.1**	93.3	88.9	91.0
+ Head POS	* **88.6**	94.4	90.1	* **92.2**
+ Verb cluster	**88.1**	94.1	89.0	**91.5**
+ Partial path	**88.2**	93.3	88.9	91.1
+ Verb sense	**88.1**	93.7	89.5	**91.5**
+ Noun head PP (only POS)	* **88.6**	94.4	90.0	* **92.2**
+ Noun head PP (only head)	* **89.8**	94.0	89.4	**91.7**
+ Noun head PP (both)	* **89.9**	94.7	90.5	* **92.6**
+ First word in constituent	* **89.0**	94.4	91.1	* **92.7**
+ Last word in constituent	* **89.4**	93.8	89.4	**91.6**
+ First POS in constituent	**88.4**	94.4	90.6	* **92.5**
+ Last POS in constituent	**88.3**	93.6	89.1	91.3
+ Ordinal const. pos. concat.	87.7	93.7	89.2	**91.4**
+ Const. tree distance	**88.0**	93.7	89.5	**91.5**
+ Parent constituent	**87.9**	94.2	90.2	* **92.2**
+ Parent head	85.8	94.2	90.5	* **92.3**
+ Parent head POS	* **88.5**	94.3	90.3	* **92.3**
+ Right sibling constituent	**87.9**	94.0	89.9	**91.9**
+ Right sibling head	**87.9**	94.4	89.9	* **92.1**
+ Right sibling head POS	**88.1**	94.1	89.9	**92.0**
+ Left sibling constituent	* **88.6**	93.6	89.6	**91.6**
+ Left sibling head	86.9	93.9	86.1	89.9
+ Left sibling head POS	* **88.8**	93.5	89.3	**91.4**
+ Temporal cue words	* **88.6**	-	-	-
+ Dynamic class context	**88.4**	-	-	-

Feature Performance

- All features are not equally useful in each task.
- Some features add more noise than information in one context than in another.
- Features can vary in efficacy depending on the classification paradigm in which they are used.
- The table in the following slide shows the effect each feature has on the argument classification and argument identification tasks when added individually to the baseline.
- Addition of named entities to the null/non-null classifier degrade the performance in this particular configuration of classifier and features.

The reason for this is the combination of two things:

- A significant number of constituents contain name entities but are not arguments of a predicate resulting in a noisy feature for null/nonnull classification.
- SVMs don't seem to handle irrelevant features very well.

FEATURE SALIENCE

- In analyzing the performance of the system, it is useful to estimate the relative contribution of the various feature sets used.
- The table in the next slide shows argument classification accuracies for combinations of features on the training and test set for all PropBank arguments using Treebank parsers.
- The features are arranged in the order of increasing salience.
- Removing all head word-related information has the most detrimental effect on performance.

The table in the following slide shows the feature salience on the task of argument identification.

- In argument classification task removing the path has the least effect on performance.

In argument identification task removing the path causes the convergence in SVM training to be very slow and has the most detrimental effect on performance.

FEATURES	ACCURACY
All features [120]	91.0
All except Path	90.8
All except Phrase Type	90.8
All except HW and HW-POS	90.7
All except All Phrases	*83.6
All except Predicate	*82.4
All except HW and FW and LW info.	*75.1
Only Path and Predicate	74.4
Only Path and Phrase Type	47.2
Only Head Word	37.7
Only Path	28.0

Table 4–10. Performance of various feature combinations on the task of argument identification

FEATURES	P	R	F_1
All features [120]	95.2	92.5	93.8
All except HW	95.1	92.3	93.7
All except Predicate	94.5	91.9	93.2
All except HW and FW and LW info.	91.8	88.5	*90.1
All except Path and Partial Path	88.4	88.9	*88.6
Only Path and HW	88.5	84.3	86.3
Only Path and Predicate	89.3	81.2	85.1

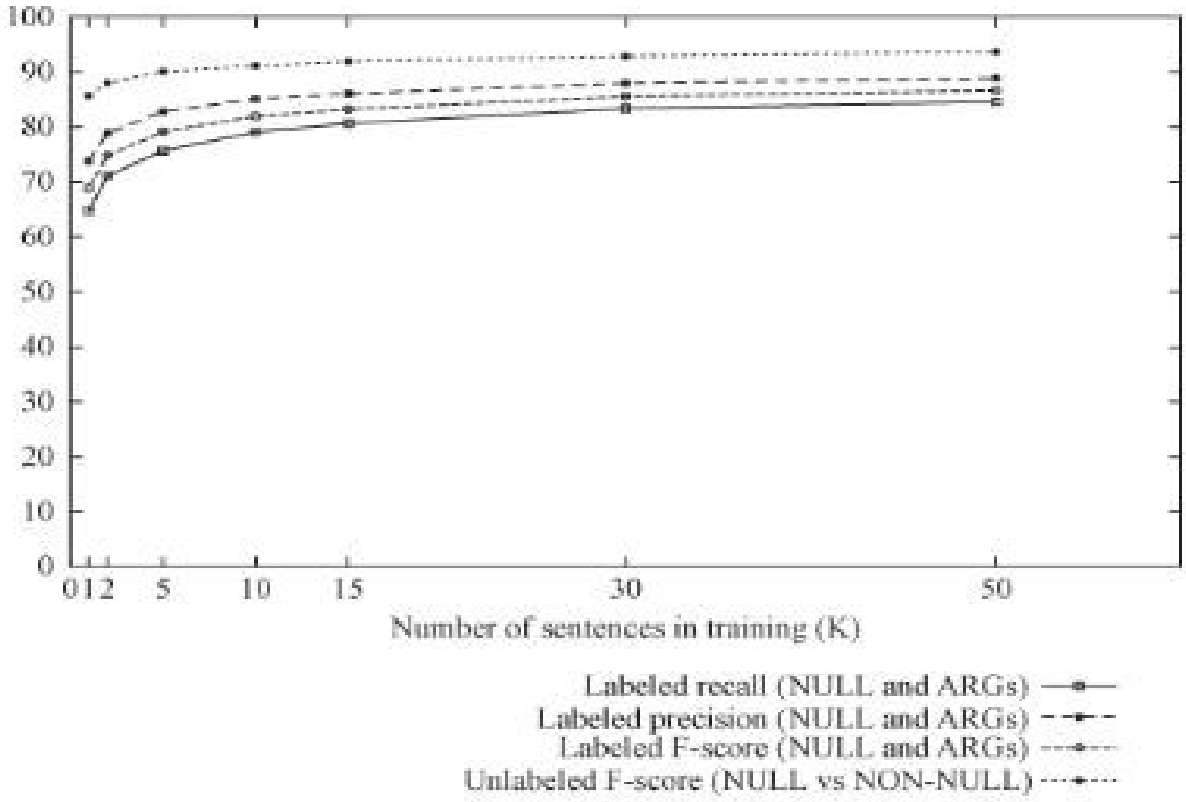
Feature Selection

- The fact that adding the named entity feature to the null/non-null classifier has a effect on the performance of the argument identification task.
- The same feature set showed significant improvement to the argument classification task.
- This indicates that a feature selection strategy would be very useful.
- One strategy is to leave one feature at a time and check the performance.
- Depending on the performance the feature is kept or pruned out.
- One more solution for Feature Selection is to convert the scores that are out put by SVMs to convert into probabilities by fitting a sigmoid.

- The probabilities resulting from either conversion may not properly calibrate.

In such case the probabilities can be binned and a warping function can be trained to calibrate tehm

Size of Training Data



An important concern in any supervised learning method is the amount of training examples required for decent performance of a classifier

- The results of classifiers trained on varying amount of training data is shown in the following figure.

- The first curve from the top indicates the change in F1 score on the task of argument identification alone.
- The third curve indicates the F1 score on the combined task of argument identification and classification.
- We can see that after 10000 examples the performance starts to plateau which indicates that simply tagging more data may not be a good strategy.
- A better strategy is to tag only appropriate new data.

The fact that both the first and the third curves run parallel to each other tells us that constant loss occurs due to classification errors through the data range.

Overcoming Parsing Errors

- After a detailed error analysis it was found that the identification problem poses a significant bottleneck to improving overall system performance.
- The baseline system's accuracy on the task of labeling nodes known to represent semantic arguments is 90%.
- Classification performance using Charnaik parses is about 3 F-score points worse than when using treebank parses.
- The severe degradation in argument identification performance for automatic parses was the motivation for examining two techniques for improving argument identification:
- Combining parses from different syntactic representations – Multiple Views
- Using n-best parses or a parse forest in the same representation- Broader Search

Multiple Views

- Pradhan and others investigated ways to combine hypotheses generated from semantic role taggers trained using different syntactic views:
- One trained using Charniak parser
- One on a rule-based dependency parser – Minipar
- One based on a flat shallow syntactic chunk representation
- They showed that these three views complement each other to improve performance.
- Although the chunk-based systems are very efficient and robust, the systems that use features based on full syntactic parses are generally more accurate.

- The syntactic parser did not produce any constituent that corresponded to the correct segmentation for the semantic argument.
- Pradhan and others report on a first attempt to overcome this problem by combining semantic role labels produced from different syntactic parses.
- They used features from the Charniak parser, the Minipar parser and a chunk-based parser.
- The main contribution of combining both the Minipar based and the Charniak-based semantic role labeler was improvement on ARG1 in addition to improvement on other arguments as shown
- The general framework is to train separate semantic role labeling systems for each of the parse tree views
- It then uses the role arguments output by these systems as additional features in a semantic role classifier using a flat syntactic view

.

An n-fold cross-validation paradigm is used to train the constituentbased role classifier and the chunk based classifier as in the figure

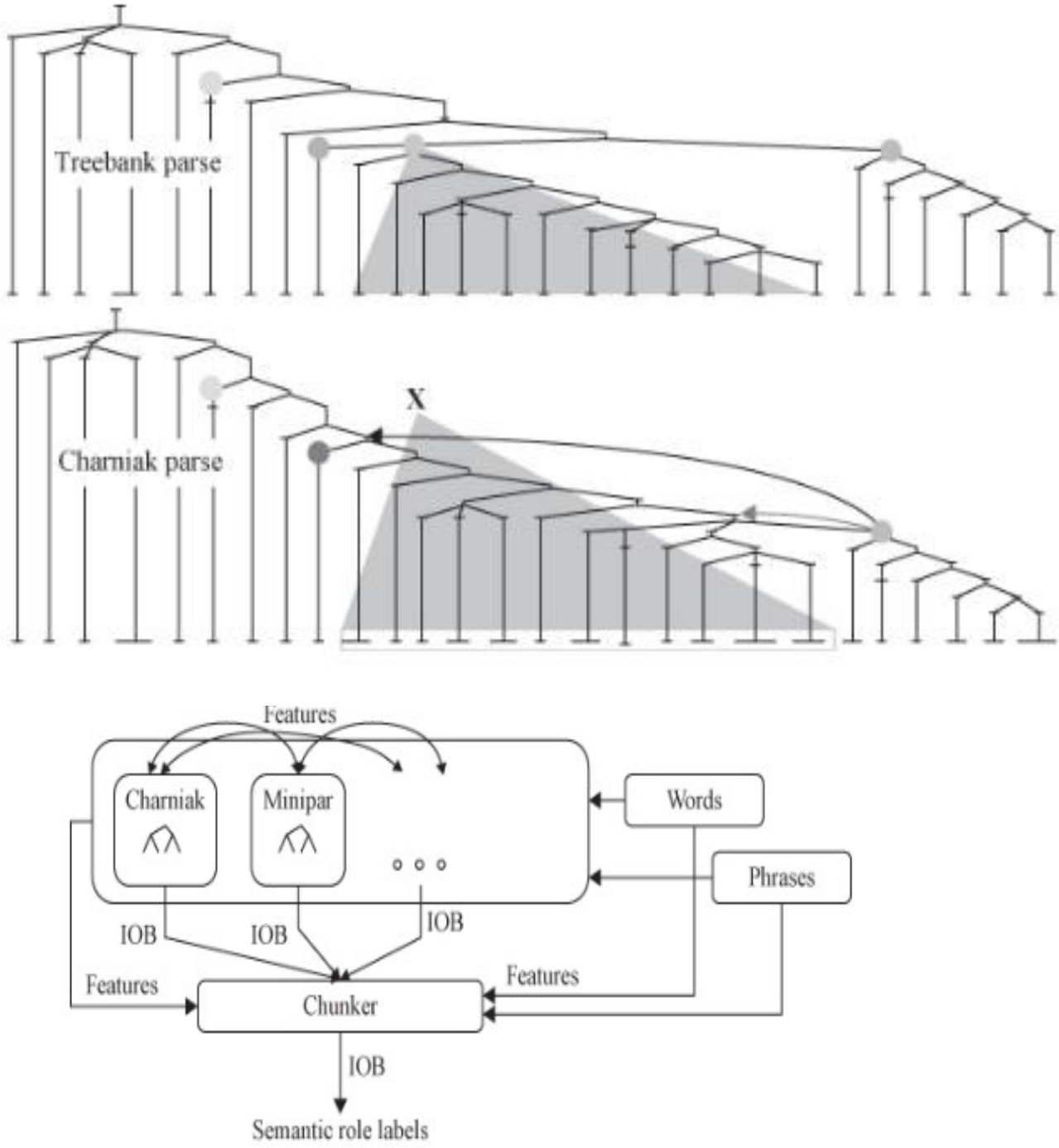


Figure 4–17. New architecture

Broader Search

- One more approach is to broaden the search by selecting constituents in n-best parses or using a packed forest representation which more efficiently represents variations over much larger n.
- Using a parse forest shows an absolute improvement of 1.2 points over single best parses and 0.5 points over n-best parses.

NOUN ARGUMENTS

- Intervening Verb Features
- Predicate NP expansion rule
- Is predicate plural
- Genitives in constituent
- Verb dominating predicate
- **Intervening Verb Features**: Support verbs play an important role in realizing the arguments of nominal predicates. Three classes of intervening verbs are used:
- Verbs of being
- Light verbs (a small set of verbs such as make, take, have) • Other verbs with part of speech starting with the string VB.
- Three features were added for each:
- A binary feature indicating the presence of the verb between the predicate and the constituent.
- The actual word as a feature
- The path through the tree from the constituent to the verb
- The following example illustrates the intuition behind these intervening verb features:

- [$_{\text{Speaker}}$ Leapor] makes general [$_{\text{Predicate}}$ assertions] [$_{\text{Topic}}$ about marriage]

- **Predicate NP expansion rule**: This is the noun equivalent of the verb subcategorization feature used by Gildea and Jurafsky. It represents the expansion rule instantiated by the syntactic parser for the lowermost NP in the tree, encompassing the predicate. This feature tends to cluster noun phrases with a similar internal structure and thus helps find argument modifiers.
- **Is predicate plural**: This binary feature indicates whether the predicate is singular or plural, as these tend to have different argument selection properties.
- **Genitives in constituent**: This is a binary feature that is true if there is a genitive word in the constituent, as these tend to be subject/object makers for nominal arguments. The following example helps clarify this notion:

- [Speaker Burma's] [Phenomenon oil] [Predicate search] hits virgin forests.

- **Verb dominating predicate**: The head word of the first VP ancestor of the predicate.

Multilingual Issues

- Since early research on semantic role labeling was performed on English corpora features and learning mechanisms were explored for English.
- Some special cases of language specific features of other languages proved to be important for the improvement of English systems. Ex: Predicate frame feature introduced for Chinese.
- Some features are language specific and have no parallels in English.
- Special word segmentation models have to be trained in the case of Chinese before parsing or semantic role labeling can begin.
- The morphology poor nature of Chinese blurs the difference between verbs, nouns, and adjectives form a closer connection between the predicates and their arguments.

- Although a similar set of features are useful across languages the specific instantiation of some can differ greatly.
- A particular characteristic of Arabic is its morphological richness.
- There are more syntactic POS categories for Arabic than there are for English or Chinese.

Unlike English Chinese and Arabic require the training of special models to identify dropped subjects before the predicate-argument structure can be fully realizedRobustness across Genre

- One important problem with all these approaches is that all the parsers are trained on the same Penn Treebank which when evaluated on sources other than WSJ seems to degrade in performance.
- It has been proved that when we train the system on WSJ data and test on the Brown propositions the classification performance and the identification performance are affected to the same degree.
- This shows that more lexical semantic features are needed to bridge the performance gap across genres.

Zapirain showed that incorporating features based on selection prefferences provide one way of effecting more lexico-semantic generalization

Software

- Following is a list of software packages available for semantic role labeling

- **ASSERT** (Automatic Statistical Semantic Role Tagger): A semantic role labeler trained on the English PropBank data.
- **C-ASSERT**: An extension of ASSERT for the Chinese language • **SwiRL**: One more semantic role labeler trained on PropBank data.
- **Shalmaneser** (A Shallow Semantic Parser): A toolchain for shallow semantic parsing based on the FrameNet data.

Meaning Representation

- Resources
- Systems
- Software
- Now we look at the activity which takes natural language input and transforms it into an unambiguous representation that a machine can act on.
- This form will be understood by the machines more than human beings.
- It is easier to generate such a form for programming languages as they impose syntactic and semantic restrictions on the programs where as such restrictions cannot be imposed on natural language.
- Techniques developed so far work within specific domains and are not scalable.
- This is called deep semantic parsing as opposed to shallow semantic parsing.

<u>Resources</u>

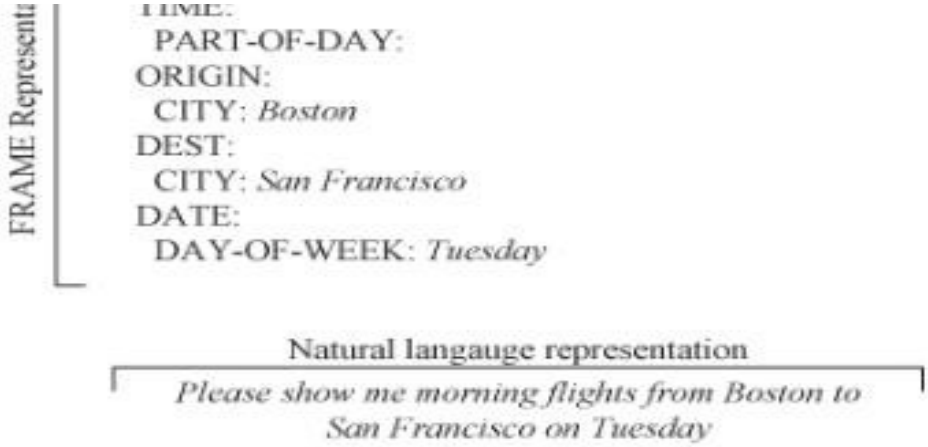
- A number of projects have created representations and resources that have promoted experimentation in this area. A few of them are as follows:
- ATIS
- Communicator

- GeoQuery
- Robocup:CLang

ATIS

- The Air Travel Information System (ATIS) is one of the first concerted efforts to build systems which build systems to transform natural language into applications to make decisions.
- Here a user query in speech is transformed using a restricted vocabulary about flight information.
- It then formed a representation that was compiled into a SQL query to extract answers from a database.

  A hierarchical frame representation was used to encode the intermediate semantic information



- The training corpus of this system includes 774 scenarios completed by 137 people yielding a total of over 7300 utterances.
- 2900 of then have been categorized and annotated with canonical reference answers.

- 600 of these have also been treebanked.

Communicator

- ATIS was more focus on user-initiated dialog Communicator involved a mixed-initiative dialog.
- Humans and machines were able to have a dialog with each other and the computer was able to present users real time information helping them negotiate a preferred itinerary.
- Many thousands of dialogs were collected and are available through the Linguistic Data Consortium.
- A lot of data was collected and annotated with dialog acts by CarnegieMellon university.

GeoQuery

- A geographical database called Geobase has about 800 Prolog facts stored in a relational databse.
- It has geographic information such as population, neighboring states, major rivers, and major cities.
- A few queries and their representations are as follows:
- What is the capital of the state with the largest population?
- Answer(C,(capital(S,C),largest(P,(state(S),population(S,P)))).
- What are the major cities in Kansas?
- Answer(C,
  (major(C),city(C),loc(C,S),equal(S,stated(kansas))))
- The Ge0Query corpus has also been translated into Japanese, Spanish and Turkish.

Robocup:CLang

- RoboCup is an international initiative by the artificial intelligence community that uses robotic soccer as its domain.
- A special formal language Clang is used to encode the advice from the team coach.

- The behaviors are expressed as if-then rules.
- Example:
- If the ball is in our penalty area, all our players except player 4 should stay in our half.
- ((bpos(penalty-area our))(do(player-except our 4)(pos(half our))))

Systems

- Depending on the consuming application the meaning representation can be a SQL query, a Prolog query, or a domain-specific query representation.
- We now look at various ways the problem of mapping the natural language to meaning representation has been tackled.

- Rule Baes
- Supervised

Rule Based

- A few semantic parsing systems that performed very well for both ATIS and Communicator projects were rule-based systems.
- They used an interpreter whose semantic grammar was handcrafted to be robust to speech recognition errors.
- Syntactic explanation of a sentence is much more complex than the underlying semantic information.
- Parsing the meaning units in the sentence into semantics proved to be a better approach.
- In dealing with spontaneous speech the system has to account for ungrammatical instructions, stutters, filled pauses etc.

Rule Based

- Word order becomes less important which leads to meaning units scattered in the sentences and not necessarily in the order that would make sense to a syntactic parser.
- Ward's system, Phoenix uses a recursive transition networks (RTNs) and a handcrafted grammar to extract a hierarchical frame structure.
- It reevaluates and adjusts the values of the frames with each new piece of information obtained.
- The system had the following error rates:
- 13.2% for spontaneous speech input of which
- 4.4% speech recognition word-error rate
- 9.3% error for transcript input

SUPERVISED

- The following are the few problems with rule-based systems:
- They need some effort upfront to create the rules
- The time and specificity required to write rules restrict the development to systems that operate in limited domains
- They are hard to maintain and scale up as the problems become more complex and more domain independent
- They tend to be brittle
- As an alternative statistical models derived from hand annotated data can be used.
- Unless some hand annotated data is available statistical models cannot deal with unknown phenomena.
- During the ATIS evaluations some data was hand-tagged for semantic information.
- Schwartz used that information to create the first end-to-end supervised statistical learning system for ATIS domain.
- They had four components in their system:
- Semantic parse

- Semantic frame
- Discourse
- Backend
- This system used a supervised learning approach with quick training augmentation through a human in-the-loop corrective approach to generate lower quality but more data for improved supervision.
- This research is now known as natural language interface for databases (NLIDB).
- Zelle and Mooney tackled the task of retrieving answers from Prolog database.
- The system tackled the task of retrieving answers from a Prolog database by converting natural language questions into Prolog queries in the domain of GeoQuery.
- The CHILL (Constructive Heuristics Induction for Language Learning) system uses a shift-reduce parser to map the input sentence into parses expressed as a Prolog program.
- A representation closer to formal logic than SQL is preferred for CHILL because it can be translated into other equivalent representations.
- It took CHILL 175 training queries to match the performance of Geobase.
- After the advances in machine learning new approaches were identified and existing were refined.

- The SCISSOR (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representation) system uses a statistical syntactic parser to create a Semantically Augmented Parse Tree (SAPT).
- Training for SCISSOR consists of a (natural language, SAPT, meaning representation) triplet.
- KRISP (Kernel-based Robust Interpretation for Semantic Parsing) uses string kernels and SVMs to improve the underlying learning techniques.

- WASP (Word Alignment based Semantic Parsing) takes a radical approach to semantic parsing by using state-of-the-art machine translation techniques to learn a semantic parser.
- Wong and Mooney treat the meaning representation language as an alternative form of natural language.
- They used GIZA++ to produce an alignment between the natural language and a variation of the meaning representation language.
- Complete meaning representations are then formed by combining these aligned strings using a synchronous CFG framework.
- SCISSOR is more accurate than WASP and KRISP, which benefits from

  SAPTs.
- These systems also have semantic parsers for Spanish, Turkish, and Japanese with similar accuracies.

Another approach is from Zettlemoyer and Collins.

- They trained a structured classifier for natural language interfaces.
- The system learned probabilistic categorical grammar (PCCG) along with a log-linear model that represents the distribution over the syntactic and semantic analysis conditioned on the natural language input.

Software

- The software programs available are as follows:
- WASP
- KRISPER
- CHILL

UNIT - V : **Discourse Processing**: Cohension, Reference Resolution, Discourse Cohension and Structure **Language Modeling**: Introduction, N-Gram Models, Language Model Evaluation, Parameter Estimation, Language Model Adaptation, Types of Language Models, Language-Specific Modeling Problems, Multilingual and Crosslingual Language Modeling

Discourse Processing

Introduction

Discourse processing in natural language processing (NLP)refers to the study and analysis of text beyond the level of individual sentences.It focuses on understanding the connections, relationships,

and coherence between sentences and larger units of text,such as paragraphs and documents.

Discourse processing aims to capture the overall meaning,structure, and flow of a piece of text, taking into account various linguistic and contextual factors.

The main goal of discourse processing is to extract meaningful information and infer the intended meaning from a text, which can be useful in a variety of NLP applications, such as text summarization, sentiment analysis, question answering, and information extraction.

It involves 4 subtasks, including:

1.Coherence and cohesion analysis:

2.Discourse structure analysis:

3.Rhetorical parsing:

4.Discourse-level sentiment analysis:

1. Coherence and cohesion analysis:

This involves examining the relationships between sentences and identifying how they connect to form a coherent and cohesive text. It includes identifying discourse markers (e.g., "however,"

"therefore"), and coreference resolution (e.g., identifying pronouns and their referents), and lexical cohesion (e.g.,identifying related words or concepts across sentences).

Example:

Text: "John loves hiking. He often goes to the mountains. The fresh air and beautiful scenery make him feel alive."

Coherence and cohesion analysis identifies that "He" in the second sentence refers to "John" in the first sentence through coreference resolution.

2.Discourse structure analysis:

This involves analyzing the hierarchical structure of a text to determine its organization and how different parts relate to each other. It includes identifying discourse relations (e.g.,

cause-effect, contrast, temporal) between sentences and determining the overall discourse structure (e.g., introduction,body, conclusion).

Example:

Text: "First, we will discuss the problem. Then, we will propose a solution. Finally, we will evaluate the results."

Discourse structure analysis identifies the temporal relations between sentences and the overall structure of the text (sequence of events).

3.Rhetorical parsing:

This involves identifying rhetorical devices and patterns used in a text, such as argumentation, persuasion, or narrative techniques.

It helps in understanding the author's intent and the overall rhetorical structure of the text.

Example:

Text: "On the one hand, reducing taxes can stimulate economic growth. On the other hand, it can lead to a decrease in government revenue. Therefore, a careful balance is necessary."

Rhetorical parsing identifies the use of contrastive rhetoric (on the one hand, on the other hand) to present different perspectives and the subsequent conclusion.

4.Discourse-level sentiment analysis:

This involves determining the sentiment or attitude expressed at the discourse level, considering the overall context and flow of the text.

It helps in understanding the overall sentiment of a document or identifying shifts in sentiment.

Example:

Text: "The movie started off slow, but it quickly picked up pace. However, the ending was disappointing."

Discourse-level sentiment analysis takes into account the overall sentiment of the text and identifies a mixed sentiment (positive at the beginning, negative at the end).

Discourse processing techniques often utilize machine learning algorithms, linguistic rules, and semantic representations to model and analyze the relationships between sentences.

They help in capturing the global structure and coherence of text, enabling deeper understanding and interpretation.

Cohesion:

In NLP, cohesion refers to the linguistic devices and techniques used to create a sense of unity and

connectedness within a text.

It involves the explicit and implicit relationships between words, phrases, and sentences that contribute to the overall coherence of the text.

Cohesion ensures that the different parts of a text are linked together logically and smoothly, allowing readers or language processing systems to understand the flow of information.

There are 5 types of cohesion that can be found in a text:

1.Reference cohesion: It involves the use of words or expressions to refer back to previously mentioned entities or ideas. This can include pronouns, demonstratives, or definite/indefinite articles.

Example:

Text: "John bought a new car. It is red and very fast."

In this example, "It" refers back to the previously mentioned car, creating reference cohesion.

2.Substitution cohesion:

It occurs when a word or phrase is substituted by another word or phrase to avoid repetition.

Example:

Text: "John likes swimming, and Mary does too."

In this example, "does too" substitutes the repetition of "likes swimming" in the second part of the sentence.

3.Ellipsis cohesion:

It involves the omission of words or phrases that can be inferred from the context.

Example:

Text: "John went to the store, and Mary to the library."

In this example, the verb "went" is omitted in the second part of the sentence, but it can be inferred from the previous context.

4.Lexical cohesion:

It is based on the use of related words or synonyms across sentences or paragraphs.

Example:

Text: "The weather was hot. The sun was shining brightly. People were enjoying the beach."

In this example, the words "weather," "sun," and "beach" are used to establish lexical cohesion.

5.Conjunction cohesion:

It involves the use of conjunctions or connectors to link sentences or ideas together.

Example:

Text: "I bought some groceries. In addition, I need to do laundry."

In this example, the conjunction "In addition" establishes cohesion between the two sentences.

Cohesion plays a crucial role in enhancing the clarity and understanding of a text.

By creating connections between different parts of a text, cohesion helps readers or language processing systems comprehend the relationships and follow the flow of information smoothly.

Reference Resolution

Reference resolution in natural language processing (NLP) refers to the process of identifying and connecting pronouns, definite/indefinite articles, or other referring expressions to their respective referents in the text.

It involves determining what entity or concept a pronoun or reference refers to in order to establish a coherent and cohesive understanding of the text.

Reference resolution is a challenging task because it requires understanding the context and identifying the correct antecedent or referent for a given expression.

The resolution can be explicit, where the referent is mentioned explicitly in the text, or implicit, where it relies on contextual information and background knowledge.

Here are a few examples to illustrate reference resolution:

Pronoun reference resolution:

Text: "John saw a dog in the park. It was chasing a ball."

In this example, the pronoun "It" refers to the previouslymentioned noun "dog." Reference resolution identifies the antecedent and connects the pronoun to its referent.

Definite article reference resolution:

Text: "I bought a book. The book is very interesting."

Here, the definite article "The" refers back to the noun "book" mentioned earlier in the text. Reference resolution connects the definite article to its referent.

Indefinite article reference resolution:

Text: "I saw a car accident. An ambulance arrived at the scene quickly."

In this example, the indefinite article "An" introduces a new entity, an ambulance, which is the referent of the article.

Demonstrative reference resolution:

Text: "This is a beautiful painting. That one is even more impressive."

Here, the demonstratives "This" and "That" establish reference to different paintings. Reference resolution identifies the specific referents based on the spatial or contextual information.

Coreference resolution:

Text: "John met Mary. He gave her a gift."

In this example, coreference resolution connects the pronouns "He" and "her" to their respective antecedents, "John" and "Mary," to establish the reference between them.

Reference resolution is a critical component in various NLP applications, such as question answering, summarization, machine translation, and information extraction.

It helps in understanding the relationships between entities and concepts in a text and enables the construction of a coherent and meaningful representation of the information.

Discourse Coherence and Structure

Discourse coherence and structure in NLP refer to the organization, flow, and logical connections between sentences and larger units of text.

It focuses on understanding how individual sentences relate to each other and contribute to the overall meaning and structure of the discourse.

Discourse coherence ensures that the text is coherent, understandable, and connected, while discourse structure deals with the arrangement and organization of different parts of the text.

Let's explore discourse coherence and structure with examples:

Discourse Coherence:

Discourse coherence involves the relationships and connections between sentences, ensuring that they form a cohesive and meaningful text.

Example:

Text: "I went to the grocery store. Bought some fruits and vegetables. The cashier was friendly."

In this example, coherence is established through the use of explicit semantic connections.

The second sentence can be understood as an elaboration of the first sentence, describing the action that occurred at the grocery store.

The third sentence introduces a new piece of information related to the grocery store visit.

Discourse Structure:

Discourse structure refers to the overall organization and arrangement of sentences, paragraphs, or sections within a text. It captures the logical and hierarchical relationships between different parts of the text.

Example:

Text: "Introduction: In this paper, we will discuss the importance of renewable energy.

Body: First, we will examine the environmental benefits. Then, we will explore the economic advantages. Finally, we will address the challenges.

Conclusion: In conclusion, renewable energy holds great potential for a sustainable future."

In this example, the text exhibits a clear discourse structure.

It starts with an introduction, followed by the body that consists of three distinct sections (environmental benefits, economic advantages, challenges), and ends with a conclusion.

The discourse structure helps readers to navigate and comprehend the content effectively.

Discourse coherence and structure can also involve explicit linguistic devices that establish connections between sentences.

Example:

Text: "John loves hiking. As a result, he spends most of his weekends exploring different trails."

In this example, the use of the discourse marker "As a result" establishes a causal relationship between the two sentences. It indicates that John's love for hiking leads to him spending

most of his weekends exploring trails.

Discourse coherence and structure are important for understanding and generating coherent and meaningful text.

They contribute to the overall readability, comprehension, and effectiveness of a written or spoken discourse, allowing for smooth information flow and logical connections between ideas.

Language Modeling

- Introduction
- n-Gram Models
- Language Model Evaluation
- Parameter Estimation
- Language Model Adaptation

- Types of Language Models
- Language Specific Modeling Problems
- Multilingual and Cross-lingual Language Modeling

Introduction

- Statistical Language Model is a model that specifies the a priori probability of a particular word sequence in the language of interest.

Given an alphabet or inventory of units $\sum$ and a sequence W= w1w2…..wt $\in \sum^*$ a language model can be used to compute the probability of W based on parameters previously estimated from a training set

- The inventory $\sum$ is the list of unique words encountered in the training data.
- Selecting the units over which a language model should be defined is a difficult problem particularly in languages other than English.

Introduction

- A language model is combined with other model or models that hypothesize possible word sequences.
- In speech recognition a speech recognizer combines acoustic model scores with language model scores to decode spoken word sequences from an acoustic signal.
- Language models have also become a standard tool in information retrieval, authorship identification, and document classification.

n-Gram Models

- Finding the probability of a word sequence of arbitrary length is not possible in natural language because natural language permits infinite number of word sequences of variable length.
- The probability P(W) can be decomposed into a product of component probabilities according to the chain rule of probability:

$$P(W) = P(w_1 \dots w_t) = P(w_1) \prod_{i=1} P(w_i | w_{i-1} w_{i-2} \dots w_2 w_1)$$

- Since the individual terms in the above product are difficult to compute directly n-gram approximation was introduced.

n-Gram Models

- The assumption is that all the preceding words except the n-1 words directly preceding the current word are irrelevant for predicting the current word.
- Hence P(W) is approximated to:

$$P(W) \approx \prod_{i=1} P(w_i | w_{i-1}, \dots w_{i-n+1})$$

- This model is also called as (n-1)-th order Markov model because of the assumption of the independence of the current word given all the words except for the n-1 preceding words.

- **Language Model Evaluation**
- model.

$$PPL(p,q) = 2^{H(p,q)} = 2^{-\sum_{i=1}^{t} p(w_i) log_2 q(w_i)}$$

- The question is how can we tell whether the language model is successful at estimating the word sequence probabilities?
- Two criteria are used:
- Coverage rate and perplexity on a held out test set that does not form part of the training data.
- The coverage rate measures the percentage of n-grams in the test set that are represented in the language model.

$$2^{-\frac{1}{t}\sum_{i=1}^{t} log_2 q(w_i)}$$

- A special case is the out-of-vocabulary rate (OOV) which is the percentage of unique word types not covered by the language model.
- The second criterion perplexity is an information theoretic measure.
- Given a model p of a discrete probability distribution, perplexity can be defined as 2 raised to the entropy of p:

$$PPL(p) = 2^{H(p)} = 2^{-\sum_x p(x) log_2 p(x)}$$

- In language modeling we are more interested in the performance of a language model q on a test set of a fixed size, say t words (w1w2wt).
- The language model perplexity can be computed as:
- q(wi) computes the probability of the ith word.
- If q(wi) is an n-gram probability, the equation becomes

$$2^{-\frac{1}{t}\sum_{i=1}^{t} log_2 q(w_i)}$$

$$2^{-\frac{1}{t}\sum_{i=1}^{t} log_2 p(w_i|w_{i-1},\ldots,w_{i-n+1})}$$

- When comparing different language models, their perplexities must be normalized with respect to the same number of units in order to obtain a meaningful comparison.
- Perplexity is the average number of equally likely successor words when transitioning from one position in the word string to the next.
- If the model has no predictive power, perplexity is equal to the vocabulary size.
- A model achieving perfect prediction has a perplexity of one.

- The goal in language model development is to minimize the perplexity on a held-out data set representative of the domain of interest.
- Sometimes the goal of language modeling might be to distinguish between “good” and “bad” word sequences.

  Optimization in such cases may not be minimizing the perplexity

Parameter Estimation

- Maximum-Likelihood Estimation and Smoothing
- Bayesian Parameter Estimation
- Large-Scale Language Models

Maximum-Likelihood Estimation and Smoothing

- The standard procedure in training n-gram models is to estimate ngram probabilities using the maximum-likelihood criterion in combination with parameter smoothing.

The maximum-likelihood estimate is obtained by simply computing

$$P(w_i|w_{i-1},w_{i-2}) = \frac{c(w_i,w_{i-1},w_{i-2})}{c(w_{i-1},w_{i-2})}$$

- relative frequencies:
- Where c(wi,wi-1,wi-2) is the count of the trigram wi-2wi-1wi in the training data.

Smoothing

- This method fails to assign nonzero probabilities to word sequences that have not been observed in the training data.
- The probability of sequences that were observed might also be overestimated.

The process of redistributing probability mass such that peaks in the n-gram probability distribution are flattened and zero estimates are floored to some small nonzero value is called smoothing

- The most common smoothing technique is **backoff**.
- Backoff involves splitting n-grams into those whose counts in the training data fall below a predetermined threshold τ and those whose counts exceed the threshold.
- In the former case the maximum-likelihood estimate of the n-gram probability is replaced with an estimate derived from the probability of the lower-order (n-1)-gram and a backoff weight.

In the later case, n-grams retain their maximum-likelihood estimates, discounted by a factor that redistributes probability mass to the lower-order distribution

$$\alpha(w_{i-1}, w_{i-2}) = \frac{1 - \sum_{w_i : c(w_i, w_{i-1}, w_{i-2}) > \tau} d_c P(w_i | w_{i-1}, w_{i-2})}{\sum_{w_i : c(w_i, w_{i-1}, w_{i-2}) \leq \tau} P_{BO}(w_i | w_{i-1})}$$

- The back-off probability P_{BO} for wi given wi-1,wi-2 is computed as follows:

$$P_{BO}(w_i | w_{i-1}, w_{i-2}) = \begin{cases} d_c P(w_i | w_{i-1}, w_{i-2}) \text{ if } c > \tau \\ \alpha(w_{i-1}, w_{i-2}) P_{BO}(w_i | w_{i-1}) \text{ otherwise} \end{cases}$$

- Where c is the count of (wi,wi-1,wi-2), and dc is a discounting factor that is applied to the higher order distribution.

- The normalization factor α(wi-1,wi-2) ensures that the entire distribution sums to one and is computed as:
    - The way in which the discounting factor is computed determines the precise smoothing technique.
    - Well-known techniques include:
    - Good-Turing
    - Written-Bell
    - Kneser-Ney
    - In Kneser-Ney smoothing a fixed discounting parameter D is applied to the raw n-gram counts before computing the probability estimates:

In modified Kneser-Ney smoothing, which is one of the most widely

$$P_{KN}(w_i|w_{i-1},w_{i-2}) = \begin{cases} \frac{max\{c(w_i,w_{i-1},w_{i-2})-D,0\}}{\sum_{w_i} c(w_i,w_{i-1},w_{i-2})} \text{ if } c > \tau \\ \alpha(w_{i-1},w_{i-2}) P_{KN}(w_i|w_{i-1}) \text{ otherwise} \end{cases}$$

used techniques, different discounting factors D1,D2,D3+ are used for n-grams with exactly one, two, or three or more counts

$$D_1 = 1 - 2Y\frac{n_2}{n_1}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2}$$

$$D_{3+} = 3 - 4Y\frac{n_4}{n_3}$$

- Where n1,n2,….. are the counts of n-grams with one, two, …, counts.
- Another common way of smoothing language model estimates is linear model interpolation.
- In linear interpolation, M models are combined by

$$P(w_i|w_{i-1}, w_{i-2}) = \sum_{m=1}^{M} \lambda_m P(w_i|h_m)$$

- Where λ is a model-specific weight.
- The following constraints hold for the model weights: 0<= λ<=1 and ∑m λm =1.

  Weights are estimated by maximizing the log-likelihood on a held-out data set that is different from the training set for the component models

This is done using the expectation-maximization (EM) procedure.

- **Bayesian Parameter Estimation**
- This is an alternative parameter estimation method where the set of parameters are viewed as a random variable governed by a prior statistical distribution.
- Given a training sample S and a set of parameters θ, $P(\theta)$ denotes a prior distribution over different possible values of θ, and $P(\theta/S)$ is the posterior distribution and is expressed using Baye's rule as:

$$P(\theta|S) = \frac{P(S|\theta)P(\theta)}{P(S)}$$

- In language modeling, $\theta = < \theta_1, \theta_2, \ldots, \theta(\theta_K) >$ (where K is the vocabulary size) for a unigram model.

- For an n-gram model θ=<P(W1/h1),…,P(Wk/hk)> with K n-grams and history h of a specified length.
- The training sample S is a sequence of words, W1…..Wt.
- We require a point estimate of θ given the constraints expressed by the prior distribution and the training sample.
- A maximum a posterior (MAP) can be used to do this.

The Bayesian criterion finds the expected value of θ given the sample S

$$\theta^B = E[\theta|S] = \int_\Theta \theta P(\theta|S)d\theta$$

$$= \frac{\int_\Theta \theta P(S|\theta)P(\theta)d\theta}{\int_\Theta P(S|\theta)P(\theta)d\theta}$$

- Assuming that the prior distribution is a uniform distribution, the MAP is equivalent to the maximum-likelihood estimate.
- Bayesian estimate is equivalent to the maximum-likelihood estimate with Laplace smoothing:

$$\theta^{MAP} = \underset{\theta\in\Theta}{\operatorname{argmax}}\, P(\theta|S) = \underset{\theta\in\Theta}{\operatorname{argmax}}\, P(S|\theta)P(\theta)$$

$$\theta_w^B = \frac{c(w)+1}{\sum_w c(w)+K}$$

- Different choices for the prior distribution lead to different estimation functions.
- The most commonly used prior distribution in language model is the Dirichlet distribution.

The Dirichlet distribution is the conjugate prior to the multinomial distribution. It is defined as

$$p(\theta) = D(\alpha_1, \ldots, \alpha_K) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

- Where Γ is the gamma function and α1,….. Ak are the parameters of the Dirichlet distribution.
- It can also be thought of as counts derived from an a priori training sample.
- The MAP estimate under the Dirichlet prior is:

$$\theta^{MAP} = \underset{\theta \in \Theta}{\operatorname{argmax}} \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{n_k + \alpha_k - 1}$$

- Where nk is the number of times word k occurs in the training sample.
- The result is another Dirichlet distribution parameterized by nk+ α
- The MAP estimate of P(θ/W,α) thus is equivalent to the maximum-likelihood estimate with add-m smoothing.
- mk= αk-1 that is pseudocounts of size αk-1 are added to each word count.

Large-Scale Language Models

- As the amount of available monolingual data increases daily models can be built from sets as large as several billions or trillions of words.
- Scaling language models to data sets of this size requires modifications to the ways in which language models are trained.
- There are several approaches to large-scale language modeling.
- The entire language model training data is subdivided into several partitions, and counts or probabilities derived from each partition are stored in separate physical locations.
- Distributed language modeling scales to vary large amounts of data and large vocabulary sizes and allows new data to be added dynamically without having to recompute static model parameters.
- The drawback of distributed approaches is the slow speed of networked queries.

One technique uses raw relative frequency estimate instead of a discounted probability if the n-gram count exceeds the minimum threshold (in this case 0):

$$S(w_i|w_{i-1},w_{i-2}) = \begin{cases} P(w_i|w_{i-1},w_{i-2}) \text{ if } c > 0 \\ \alpha S(w_i|w_{i-1}) \text{ otherwise} \end{cases}$$

- The α parameter is fixed for all contexts rather than being dependent on the lower-order n-gram.
- An alternative possibility is to use large-scale distributed language models at a second pass rescoring stage only, after first-pass hypotheses have been generated using a smaller language model.

The overall trend in large-scale language modeling is to abandon exact parameter estimation of the type described in favor of approximate techniques.

Language Model Adaptation

- Language model adaptation is about designing and tuning model such that it performs well on a new test set for which little equivalent training data is available.

- The most commonly used adaptation method is that of mixture language models or model interpolation.
- One popular method is topic-dependent language model adaptation.
- The documents are first clustered into a large number of different topics and individual language models can be built for each topic cluster.
- The desired final model is then fine-tuned by choosing and interpolating a smaller number of topic-specific language models.
- A form of dynamic self-adaptation of a language model is provided by trigger models.
- The idea is that in accordance with the underlying topic of the text, certain word combinations are more likely than other to co-occur.

Some words are said to trigger others for example the words stock and market in a financial news text