PBR VITS (AUTONOMOUS)

III B.TECH CSE-AI

NATURAL LANGUAGE PROCESSING

TEXT BOOK: James Allen, **Natural Language Understanding**, 2nd Edition, 2003, Pearson Education

Course Instructor: Dr KV Subbaiah, M.Tech, Ph.D, Professor, Dept. of CSE

UNIT–IV Interpretation and Modelling

Semantic Interpretation-Semantic & Logical form, Word senses & ambiguity, the basic logical form language, encoding ambiguity in the logical Form, Verbs & States in logical form, Thematic roles, Speech acts &embedded sentences, Defining semantics structure model theory.
Language Modelling-Introduction, n-Gram Models, Language model Evaluation, Parameter Estimation, Language Model Adaption, Types of Language Models, Language-Specific Modelling Problems, Multilingual and Cross lingual Language Modelling.

4.1 Semantic Interpretation-Semantic & Logical form

Precisely defining the notions of semantics and meaning is surprisingly difficult because the terms are used for several different purposes in natural and technical usage. For instance, there is a use of the verb "*mean"* that has nothing to do with language. Say you are walking in the woods and come across a campfire that is just noticeably warm. You might say "*This fire means someone camped here last night"*. By this you mean that the fire is evidence for the conclusion or implies the conclusion. This is related to, but different from, the notion of meaning that will be the focus of the next few chapters. The meaning we want is closer to the usage when defining a word, such as in the sentence *"'Amble' means to walk slowly"*. This defines the meaning of a word in terms of other words. To make this more precise, we will have to develop a more formally specified language in which we can specify meaning without having to refer back to natural language itself. But even if we can do this, defining a notion of sentence meaning is difficult. For example, I was at an airport recently and while I was walking towards my departure gate, a guard at the entrance asked, "Do you know what gate you are going to?" I interpreted this as asking whether I knew where I was going and answered yes. But this response was based on a misunderstanding of what the guard meant, as he then asked, "Which gate is it?" He clearly had wanted me to tell him the gate number. Thus the sentence "*Do you know what gate you are going to?"* appears to mean different things in different contexts.

Can we define a notion of sentence meaning that is independent of context? In other words, is there a level at which the sentence "*Do you know what gate you are going to?"* has a single meaning, but may be used for different purposes? This is a complex issue, but there are many advantages to trying to make such an approach work. The primary argument is modularity. If such a division can be made, then we can study sentence meaning in detail without all the complications of sentence usage. In particular, if sentences have no context-independent meaning, then we may not be able to separate the study of language from the study of general human reasoning and context. As you will see in the next few chapters, there are many examples of constraints based on the meaning of words that appear to be independent of context. So from now on, we will use the term "*meaning"* in this context-independent sense, and we will use the term "*usage"* for the context-dependent aspects. The representation of context-independent meaning is called the logical form. The process of mapping a sentence to its logical form is called semantic interpretation, and the process of mapping the logical form to the final knowledge representation (KR) language is called contextual interpretation. Figure 8.1 shows a simple version of the stages of interpretation. The exact meaning of the notations used will be defined later.

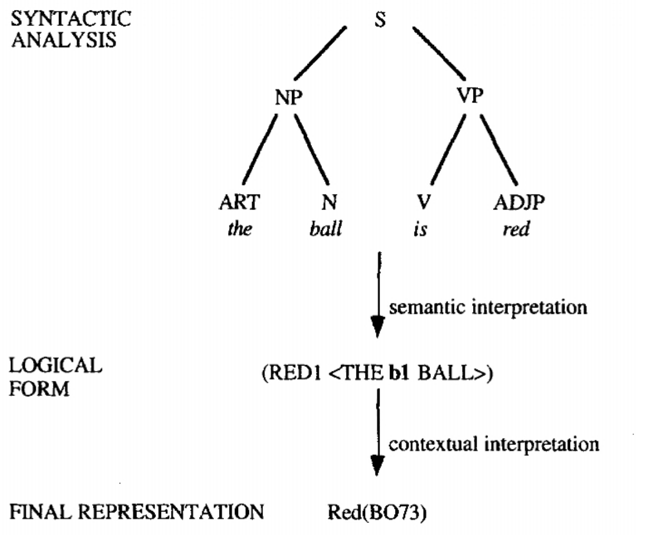


Figure 8.1 Logical form as an intermediate representation

For the moment let us assume the knowledge representation language is the first-order predicate calculus (FOPC). Given that assumption, what is the status of the logical form? In some approaches the logical form is defined as the literal meaning of the utterance, and the logical form language is the same as the final knowledge representation language. If this is to be a viable approach in the long run, however, it would mean that the knowledge representation must be considerably more complex than representations in present use in Al systems. For instance, the logical form language must allow indexical terms, that is, terms that are defined by context. The pronouns "*I"* and "*you"* are indexical because their interpretation depends on the context of who is speaking and listening. In fact most definite descriptions (such as "*the red ball")* are indexical, as the object referred to can only be identified with respect to a context. Many other aspects of language, including the interpretation of tense and determining the scope of quantifiers, depend on context as well and thus cannot be uniquely determined at the logical form level. Of course, all of this could be treated as ambiguity at the logical form level, but this would be impractical, as every sentence would have large numbers of possible logical forms (as in the sentence "*The red ball dropped"*, which would have a different logical form for every possible object that could be described as a ball that is red).
But if the logical form language is not part of the knowledge representation language, what is its formal status? A promising approach has been developed in linguistics over the last decade that

suggests an answer that uses the intuitive notion of the meaning of "*situation"* in English. For instance, when attending a class, you are in a situation where there are fellow students and an instructor, where certain utterances are made by the lecturer, questions asked, and so on. Also, there will be objects in the lecture hail, say a blackboard and chairs, and so on. More formally, you might think of a situation as a set of objects and relations between those objects. A very simple situation might consist of two objects, a ball B0005 and a person P86, and include the relationship that the person owns the ball. Let us encode this situation as the set {(BALL B0005), (PERSON P86), (OWNS P86B0005)}.

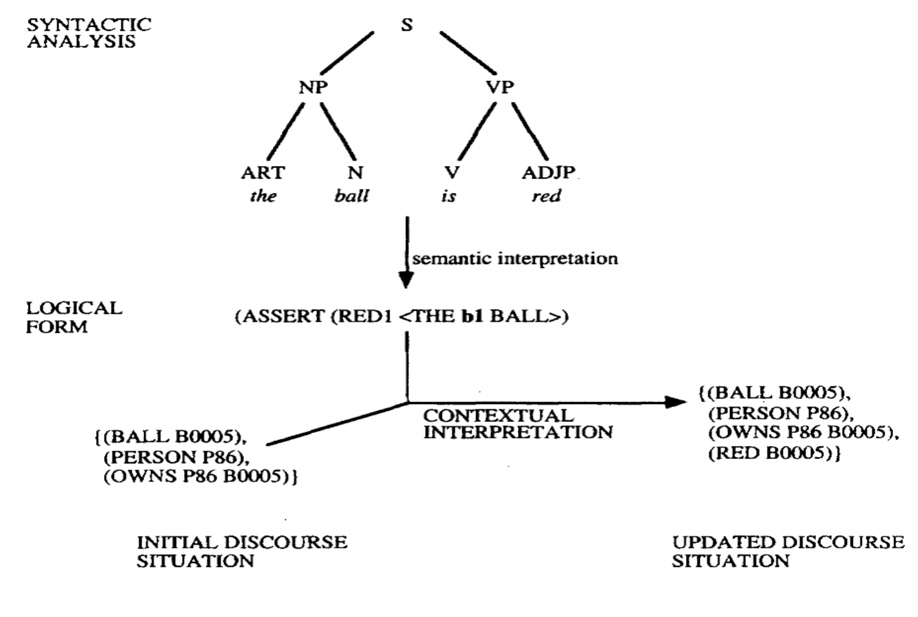

SYNTACTIC ANALYSIS
S
NP
VP
ART
N
V
ADJP
the
ball
is
red
semantic interpretation
LOGICAL FORM
(ASSERT (RED1 <THE **b1** BALL>)
{(BALL B0005), (PERSON P86), (OWNS P86 B0005)}
CONTEXTUAL INTERPRETATION
{(BALL B0005), (PERSON P86), (OWNS P86 B0005), (RED B0005)}
INITIAL DISCOURSE SITUATION
UPDATED DISCOURSE SITUATION

Figure 8.2 Logical form as a function

Language creates special types of situations based on what information is conveyed. These issues will be explored in detail later, but for now consider the following to help your intuition. In any conversation or text, assume there is a discourse situation that records the information conveyed so far. A new sentence is interpreted with respect to this situation and produces a new situation that includes the information conveyed by the new sentence. Given this view, the logical form is a function that maps the discourse situation in which the utterance was made to a new discourse situation that results from the occurrence of the utterance. For example, assume that the situation we just encoded has been created by some preceding sentences describing the ball and who owns it. The utterance "*The ball is red"* might produce a new situation that consists of the old situation plus the new fact that B0005 has the property RED: ((BALL B0005), (PERSON P86), (OWNS P86 B0005*)*, (RED B0005)}. Figure 8.2 shows this view of the interpretation process, treating the logical form as a function between situations. The two organizations presented in Figures 8.1 and 8.2 differ in that the latter might not include a single identifiable expression in the knowledge representation that fully captures the "meaning" of the sentence. Rather, the logical form might make a variety of changes to produce the updated situation. This allows other implications to be derived from an utterance that are not directly captured in the semantic content of the sentence. Such issues will become important later when we discuss contextual interpretation.

4.2 Word senses & ambiguity

To develop a theory of semantics and semantic interpretation, we need to develop a structural model, just as we did for syntax. With syntax we first introduced the notion of the basic syntactic classes and then developed ways to constrain how simple classes combine to form larger structures. We will follow the same basic strategy for semantics. You might think that the basic semantic unit could be the word or the morpheme, but that approach runs into problems because of the presence of ambiguity. For example, it is not unusual for the verb *go* to have more than 40 entries in a typical dictionary. Each one of these definitions reflects a different sense of the word. Dictionaries often give synonyms for particular word senses. For *go* you might find synonyms such as *move, depart, pass, vanish, reach, extend,* and *set out.* Many of these highlight a different sense of the verb *go.* Of course, if these are true synonyms of some sense of *go,* then the verbs themselves will share identical senses. For instance, one of the senses of *go* will be identical to one of the senses of *depart.*

If every word has one or more senses then you are looking at a very large number of senses, even given that some words have synonymous senses. Fortunately, the different senses can be organized into a set of broad classes of objects by which we classify the world. The set of different classes of objects in a representation is called its ontology. To handle a natural language, we need a much broader ontology than commonly found in work on formal logic. Such classifications of objects have been of interest for a very long time and arise in the writings of Aristotle (384—322 B.C.). The major classes that Aristotle suggested were substance (physical objects), quantity (such as numbers), quality (such as bright red), relation, place, time, position, state, action, and affection. To this list we might add other classes such as events, ideas, concepts, and plans. Two of the most influential classes are actions and events. Events are things that happen in the world and are important in many semantic theories because they provide a structure for organizing the interpretation of sentences. Actions are things that agents do, thus causing some event. Like all objects in the ontology, actions and events can be referred to by pronouns, as in the discourse fragment

We lifted the box. It was hard work.

Here, the pronoun "*it"* refers to the action of lifting the box. Another very influential category is the situation. As previously mentioned, a situation refers to some particular set of circumstances and can be viewed as subsuming the notion of events. In many cases a situation may act like an abstraction of the world over some location and time. For example, the sentence "*We laughed and sang at the football game"* describes a set of activities performed at a particular time and location, described as the situation "*the football game"*. Not surprisingly, ambiguity is a serious problem during semantic interpretation. We can define a word as being semantically ambiguous if it maps to more than one sense. But this is more complex than it might first seem, because we need to have a way to determine what the allowable senses are. For example,

4.3 The Basic Logical Form Language

The basic logical form language defines a language in which you can combine word senses elements to form meanings for more complex expressions. This language will follow **First-Order Predicative Calculus** (FOPC). The FOPC are many equivalent forms of representation, such as network-based representations, that use the same basic ideas. The word senses will serve as the atoms or constants of the representation. These constants can be classified by the types of things they describe. For instance, constants that describe objects in the world,

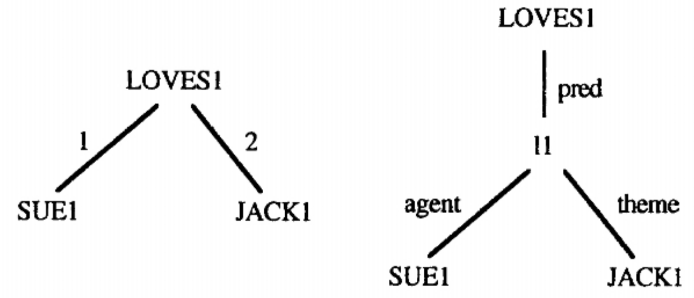


Figure 8.3 Two possible network representations of *Sue loves Jack*

including abstract objects such as events and situations, are called terms. Constants that describe relations and properties are called predicates. A proposition in the language is formed from a predicate followed by an appropriate number of terms to serve as its arguments. For instance, the proposition corresponding to the sentence "*Fido is a dog"* would be constructed from the term FIDO1 and the predicate constant DOG1 and is written as

(DOG1 FIDO1)

Predicates that take a single argument are called unary predicates or properties; those that take two arguments, such as LOVES1, are called binary predicates; and those that take *n* arguments are called n-ary predicates. The proposition corresponding to the sentence *Sue loves Jack* would involve a binary predicate LOVES1 and would be written as

(LOVES1 SUE1 JACK1)

You can see that different word classes in English correspond to different types of constants in the logical form. Proper names, such as *Jack,* have word senses that are terms; common nouns, such as *dog,* have word senses that are unary predicates; and verbs, such as *run, love,* and *put,* have word senses that correspond to *n-ary* predicates, where *n* depends on how many terms the verb subcategorizes for.

Note that while the logical forms are presented in a predicate-argument form here, the same distinctions are made in most other meaning representations. For instance, a network representation would have nodes that correspond to the word senses and arcs that indicate the predicate-argument structure. The meaning of the sentence *Sue loves Jack* in a semantic network like representation might appear in one of the two forms shown in Figure 8.3. For most purposes all of these representation formalisms are equivalent.

More complex propositions are constructed using a new class of constants called logical operators. For example, the operator NOT allows you to construct a proposition that says that some proposition is not true. The proposition corresponding to the sentence *Sue does not love Jack* would be

(NOT (LOVES1 SUE1 JACK1))

English also contains operators that combine two or more propositions to form a complex proposition. FOPC contains operators such as disjunction (v), conjunction (&), what is often called implication **(`),** and other forms (there are 16 possible truth functional binary operators in FOPC). English contains many similar operators including *or, and, if only if,* and so on. Natural language connectives often involve more complex relationships between sentences. For instance, the conjunction "*and"* might correspond to the logical operator "&" but often also involves temporal sequencing, as in "*I went home and had a drink"*, in which going home preceded having the drink. The connective "*but"*, on the other hand, is like "*and"* except that the second argument is something that the hearer might not expect to be true given the first argument. The general form for such a proposition is *(connective proposition proposition).* For example, the logical form of the sentence "*Jack loves Sue or Jack loves Mary"* would be **(OR1 (LOVES1 JACK1 SUE1) (LOVES1 JACK1 MARY1))**. The logical form language will allow both operators corresponding to word senses and operators like "&" directly from FOPC. The logic based operators will be used to connect propositions not explicitly conjoined in the sentence.

4.4 Encoding Ambiguity in the Logical Form

The basic logical form language defines a language in which you can combine word senses elements to form meanings for more complex expressions. The Logical form language will follow **First-Order Predicative Calculus** (FOPC) to represent the given corpus of the NLP.

The Need for Generalized Quantifiers:

The Quantifier is used to present more complex sentences by using FOPC and there are only two quantifiers: **Ž** and **j**. English contains a much larger range of quantifiers, including *all, some, most, many, a few, the,* and so on. Thus two instances of the same variable x occurring in two different formulas - say in the formulas **jx.P(x)** and **jx. Q(x).**

For the standard existential and universal quantifiers, there are formulas in standard FOPC equivalent to the generalized quantifier forms. In particular, the formula

(EXISTS x : Px Qx)

is equivalent to

j x . Px & Qx)

and the universally quantified form

(ALL x : Px Qx)

is equivalent to

Žx . Px ` Qx)

These generalized quantifier forms can be thought of simply as abbreviations. But the other quantifiers do not have an equivalent form in standard FOPC. To see this, consider trying to define the meaning of "*Most dogs bark"* using the standard semantics for quantifiers. Clearly **Žx . Dog(x) ` Bark(x)**is too strong, and **jx . Dog(x) ` Bark(x)** is too weak.
Thus the sentence "*Sue watched the ball"* is ambiguous out of context. A single logical form can represent these two possibilities, however:
1. (THE **b1** : ({BALL1 BALL2} **b1**) (PAST (WATCH1 SUE1 **b1**)))
This abbreviates two possible logical forms, namely
2. (THE **b1** : (BALL1 **b1**) (PAST (WATCH1 SUE1 **b1**)))
and
3. (THE **b1** : (BALL2 **b1**) (PAST (WATCH1 SUE1 **b1**)))

Encoding Ambiguity

A typical sentence will have multiple possible syntactic structures, each of which might have multiple possible logical forms.
The words in the sentence will have multiple senses. Many researchers view this ambiguity encoding as a separate level of representation from the logical form, and it is often referred to as the **quasi-logical form**.

For example, the logical forms for the sentence "Every *boy loves a dog"* are captured by a single ambiguous form
(LOVES1 <EVERY **b1** (BOY1 **b1**)> <A **d1** (DOG1 **d1**)>)
This abbreviates an ambiguity between the logical form
(EVERY **b1** : (BOY1 **b1**) (A **d1** (DOG1 **d1**) (LOVES1 **b1 d1**)))
and
(A **d1**: (DOG1 **d1**) (EVERY **b1** : (BOY1 **b1**) (LOVES1 **b1 d1**)))

In addition, operators such as negation and tense are also scope sensitive. For example, the sentence "*Every boy didn't run"* is ambiguous between the read ing in which some boys didn't run and some did, that is,
(NOT (EVERY **b1** : (BOY1 **b1**) (RUN1 **b1**)))

In fact, proper names must be interpreted in context, and the name *John* will refer to different people in different situations. We will introduce this construct as a special function, namely
(NAME <variable> <name>)
which produces the appropriate object with the name in the current context. Thus, the logical form of *"John ran"* would be **(<PAST RUN1> (NAME j1 "John")).**

4.5 Verbs & States in Logical Form

The verbs have mapped to appropriate senses acting as predicates in the logical form. This treatment can handle all the different forms but loses some generalities that could be captured. It also has some annoying properties. Consider the following sentences, all using the verb "*break":*

John broke the window with the hammer.
The hammer broke the window.
The window broke.

The verb "*break"* mapped to the same sense in each case.
The first seems to be a ternary relation between John, the window, and the hammer, the second a binary relation between the hammer and the window, and the third a unary relation involving the window. It seems you would need three different senses of break, BREAK1, BREAK2, and BREAK3, that differ in their arity and produce logical forms such as

1. (<PAST BREAK1> (NAME j1 "John") <THE w1 WINDOW1> <THE h1 HAMMER1>),
2. (<PAST BREAK2> <THE h1 HAMMER1> <THE w1 WINDOW1>), and
3. (<PAST BREAK3> <THE w1 WINDOW1>)

Furthermore, to guarantee that each predicate is interpreted appropriately, the representation would need axioms so that whenever 1 is true, 2 is true, and whenever 2 is true, 3 is true. These axioms are often called meaning postulates.

In particular, the quasi-logical form for the sentence "*John broke the window"* using this abbreviation is
(<PAST BREAK!> **e1** [AGENT (NAME **j1** "John")]
[THEME <THE **w1** WINDOW1>])
It turns out that similar arguments can be made for verbs other than event verbs. Consider the sentence "*Mary was unhappy"*. If it is represented using a unary predicate as
(<PAST UNHAPPY> (NAME **j1** "Mary"))

In many situations using explicit event and state variables in formulas is cumbersome and interferes with the development of other ideas. As a result, we will use different representations depending on what is best for the presentation. For example, the logical form of ***"Mary sees John"*** will sometimes be written as
(PRES (SEES1 **l1** [AGENT (NAME **j1** "Mary")]
[THEME (NAME **m1** "John")]))
which of course is equivalent to
(PRES (j **l1** (& (SEES1 **l1**) (AGENT **l1** (NAME **j1** "Mary"))
(THEME **l1** (NAME **m1** "John")))))

4.6 Thematic roles

Thematic Roles and theories based on the notion of thematic roles, or cases. One of the motivating example is given below.

- **John broke the window with the hammer.**
- **The hammer broke the window.**
- **The window broke.**

John", *" the hammer"*, and "*the window"* play the same semantic roles in each of these sentences. "*John"* is the actor, "*the window"* is the object, and "*the hammer"* is the instrument used in the act of breaking of the window.

We introduced relations such as AGENT, THEME, and INSTR to capture these intuitions.
Perhaps the easiest thematic role to define is the AGENT role.
A noun phrase fills the AGENT role if it describes the instigator of the action described by the sentence. The following sentences are acceptable: (bcz Agent role)

- **John intentionally broke the window.**
- **John broke the window in order to let in some air.**

But these sentences are not acceptable:
*** The hammer intentionally broke the window.**
*** The window broke in order to let in some air.**

For example, given the sentence "***The gray eagle saw the mouse***", the NP "***the mouse***" is the THEME and is the answer to the question "What was seen?"
For intransitive verbs, the THEME role is used for the subject NPs that are not AGENTs. Thus in "***The clouds appeared over the horizon***", the NP "*the clouds*" fills the THEME role. More examples follow, with the THEME NP in italics:

- ***The rock*** broke.
- John broke ***the rock***.
- I gave John ***the book***.

Other phrases describe changes in location, direction of motion, or paths:

- I walked ***from here to school*** yesterday.
- It fell ***to the ground***.
- The birds flew ***from the lake along the river gorge***.

There are at least three different types of phrases here: those that describe where something came from (the FROM-LOC role), such as "*from here";* those that describe the destination (the TO-LOC role), such as "*to the ground";* and those that describe the trajectory or path (the PATH-LOC role), such as "*along the gorge"*.

You can see other specializations of these roles when you consider the abstract relation of possession:

- **I threw the ball *to John.*** **(the TO-LOC role)**
- **I gave a book *to John.*** **(the TO-POSS role)**
- **I caught the ball *from John.*** **(the FROM-LOC role)**
- **I borrowed a book *from John.* (the FROM-POSS role)**
- ***The box* contains a ball.** **(the AT LOC role)**
- ***John* owns a book.** **(the AT POSS role)**

Similarly, you might define AT-TIME, TO-TIME, and FROM-TIME roles, as in

- **I saw the car *at 3 o'clock.*** **(the AT-TIME role)**
- **I worked *from one until three.* (the FROM-TIME and TO-TIME role)**
- **The temperature remains *at zero.*** **(AT VALUE)**
  The temperature rose *from zero.* **(FROM-VALUE)**

4.7 Speech Acts and Embedded Sentences

Sentences are used for many different purposes. Each sentential mood indicates a different relation between the speaker and the propositional content of the context. The logical form language is extended to capture the distinctions. Each of the major sentence types has a corresponding operator that takes the sentence interpretation as an argument is called a surface speech act.

They are indicated by new operators as follows:

- **ASSERT** - the proposition is being asserted.
- **Y/N-QUERY -** the proposition is being queried.
- **COMMAND** - the proposition describes an action to perform.
- **WH-QUERY** - the proposition describes an object to be identified.

For ASSERT declarative sentences, such as "***The man ate a peach***", the complete LF is (logical form)

- (ASSERT (<PAST EAT> **e1** [AGENT <THE **m1** MAN1] [THEME <A **p1** PEACH1>]))

For YES/NO questions, such as "*Did the man eat a peach?"*, the LF is

- (Y/N-QUERY (<PAST EAT> **e1** [AGENT <THE **m1** MAN1>] [THEME <A **p1** PEACH1>]))

For COMMAND, such as "*Eat the peach"*, the LF is

- (COMMAND (EAT **e1** [THEME <THE **p1** PEACH1>]))

For WH-QUERY, the logical form of the sentence "*What did the man eat?"* is

- (WH-QUERY (<PAST EAT> **e1** (AGENT <THE **m1** MAN1>] [THEME <WH **w1** PHYSOBJ>]))

Embedded sentences

Embedded sentences, such as relative clauses, end up as complex restrictions within the noun phrase construction and thus do not need any new notation. For example, the logical form of the sentence "*The man who ate a peach left*" would be

(ASSERT
(<PAST LEAVE> **l1**
[AGENT <THE **m1** (& (MAN1 **m1**)
(<PAST EAT1> **e2** [AGENT **m1**]
[THEME <A **p1** PEACH>]))>]))

4.8 Defining Semantic Structure: Model Theory

The basic building block for defining semantic properties is known as a model. A model theory can be thought of as a set of objects and their properties and relationships, together with a specification of how the language being studied relates to those objects and relationships.

A model can be thought of as representing a particular context in which a sentence is to be evaluated.

For instance, the standard models for logic, called Tarskian models, are complete in that they must map every legal term in the language into the domain and assign every statement to be true or false. Model theory is an excellent method for studying context-independent meaning, because the meanings of sentences are not defined with respect to one specific model but rather by how they relate to any possible model.

Formally, a model m is a tuple **<Dm, Im>,** where **Dm** is the domain of interpretation (that is, a set of primitive objects), and **I** is the interpretation function.

- To handle natural language, the domain of interpretation would have to allow objects of all the different types of things that can be referred to, including physical objects, times, Locations, events, and situations.
- The interpretation function maps senses and larger structures into structures defined on the domain. For example, the following describe how an interpretation function will interpret the senses based on some lexical classes:

Senses of noun phrases - refer to specific objects; the interpretation function maps each to an element of **Dm**.
Senses of singular common nouns (such as "*dog", "idea", "party")* - identify classes of objects in the domain; the interpretation function maps them to sets of elements from **Dm** (that is, subsets of **Dm**).
Senses of verbs - identify sets of n-ary relations between objects in **D**. The arity depends on the verb.

UTTERANCE → (ASSERT *PROPOSITION*) |
(Y/N-QUERY *PROPOSITION*) |
(COMMAND *PROPOSITION*) |
(WH-QUERY *PROPOSITION*)

PROPOSITION → (*n-ARY-OPERATOR* $PROPOSITION_1$ ... $PROPOSITION_n$) |
(*QUANTIFIER VARIABLE : PROPOSITION PROPOSITION*) |
(*n-ARY-PREDICATE* $TERM_1$ ... $TERM_n$) |
(*EVENT-STATE-PRED VARIABLE* $[ROLE\text{-}NAME\ TERM]_1$ ... $[ROLE\text{-}NAME\ TERM]_n$)

TERM → *VARIABLE* |
(NAME *VARIABLE NAME-STRING*) |
(PRO *VARIABLE PROPOSITION*)

1-ARY-OPERATOR → NOT | PAST | PERF | PROG | ...

2-ARY-OPERATOR → AND | BUT | IF-THEN | ...

QUANTIFIER → THE | SOME | WH | ∃ | ..

VARIABLE → **b1** | **man3** | ...

1-ARY-PREDICATE → *TYPE-PREDICATE* | HAPPY1 | RED1 | ...

TYPE-PREDICATE → *EVENT-STATE-PRED* | (PLUR *TYPE-PREDICATE*) | MAN1 | ...

EVENT-STATE-PRED → RUN1 | LOVE3 | GIVE1 | HAPPY | ...

2-ARY-PREDICATE → *ROLE-NAME* | ABOVE1 | ...

ROLE-NAME → AGENT | THEME | AT-LOC | INSTR | ...

NAME-STRING → "John" | "The New York Times" | ...

Figure 8.7 A formal definition of the syntax of the logical form language

TERM → <*QUANTIFIER VARIABLE PROPOSITION*>

TERM → <*n-ARY-OPERATOR* $TERM_1$ ... $TERM_n$>

n-ARY-PREDICATE
→ <*m-ARY-OPERATOR* $n\text{-}ARY\text{-}PREDICATE_1$ ... $n\text{-}ARY\text{-}PREDICATE_m$>

n-ARY-OPERATOR → {$n\text{-}ARY\text{-}OPERATOR_1$ ... $n\text{-}ARY\text{-}OPERATOR_m$}

QUANTIFIER → {$QUANTIFIER_1$ ... $QUANTIFIER_m$}

n-ARY-PREDICATE → {$n\text{-}ARY\text{-}PREDICATE_1$ ... $n\text{-}ARY\text{-}PREDICATE_m$}

TYPE-PREDICATE → {$TYPE\text{-}PREDICATE_1$ ... $TYPE\text{-}PREDICATE_m$}

EVENT-STATE-PRED → {$EVENT\text{-}STATE\text{-}PRED_1$ ... $EVENT\text{-}STATE\text{-}PRED_m$}

ROLE-NAME → {$ROLE\text{-}NAME_1$ ... $ROLE\text{-}NAME_m$}

Figure 8.8 Additional rules defining the quasi-logical form

4.9 Language Modelling-Introduction and n-Gram Models

The Models that assign probabilities to sequences of words are called **language models (LMs).** The simplest language model that assigns probabilities to sentences and sequences of words is the **n-gram** model.

An **n-gram**is a sequence of N words: For example "**Please turn your homework**" can be written as:
–A 1-gram (unigram) is a single word sequence of words like "please", "turn", "your", "homework"

–A 2-gram (bigram) is a two-word sequence of words like "please turn", "turn your", or "your homework".

–A 3-gram (trigram) is a three-word sequence of words like "please turn your", or "turn your homework".

We can use n-gram models to estimate the probability of the last word of an n-gram given the previous words, and also to assign probabilities to entire word sequences.

Probabilistic Language Models:

Probabilistic language models can be used to assign a probability to a sentence in many NLP tasks.

•Machine Translation: P(highwinds tonight) > P(largewinds tonight)
•Spell Correction: Thek office is about ten minutes from here
P(The Office is) > P(Then office is)
•Speech Recognition:
P(I saw a van) >> P(eyes awe of an)
•Summarization, question-answering.

Our goal is to compute the probability of a sentence or sequence of words W (=w1,w2,…wn):
P(W) = P(w1,w2,w3,w4,w5…wn)
•What is the probability of an upcoming word?: P(w5|w1,w2,w3,w4)
•A model that computes either of these: P(W) or P(wn|w1,w2…wn-1) is called a **language model**.

Chain Rule of Probability

We compute probabilities of entire word sequences like w1,w2,…wn
The probability of the word sequence w1,w2,…wn is P(w1,w2,…wn).
We can use the **chain rule of the probability** to decompose this probability:

$$\mathbf{P(w_1^n) = P(w_1)\,P(w_2|w_1)\,P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})}$$

$$= \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

Example: P(the man from jupiter) =
P(the) P(man|the) P(from|the man) P(jupiter|the man from)

N-Grams

The aim of the n-gram model (simplifying assumption) is to predict the next word in the given text. Instead of computing the probability of a word given its entire history, we can approximate the history by just the last few words.

$$P(w_n|w_1 \ldots w_{n-1}) \approx P(w_n) \qquad \textbf{unigram}$$
$$P(w_n|w_1 \ldots w_{n-1}) \approx P(w_n|w_{n-1}) \qquad \textbf{bigram}$$
$$P(w_n|w_1 \ldots w_{n-1}) \approx P(w_n|w_{n-1}w_{n-2}) \qquad \textbf{trigram}$$
$$P(w_n|w_1 \ldots w_{n-1}) \approx P(w_n|w_{n-1}w_{n-2}w_{n-3}) \qquad \textbf{4-gram}$$
$$P(w_n|w_1 \ldots w_{n-1}) \approx P(w_n|w_{n-1}w_{n-2}w_{n-3}w_{n-4}) \qquad \textbf{5-gram}$$

- In general, **N-Gram** is

$$P(w_n|w_1 \ldots w_{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

N-Grams
Computing probabilities of word sequences (Sentences)

Unigram
P(<s> the man from Jupiter came </s>)
P(the) P(man) P(from) P(jupiter) P(came)

Bigram
P(<s> the man from Jupiter came </s>)
P(the|<s>) P(man|the) P(from|man) P(jupiter|from) P(came|jupiter) P(</s>|came)

Trigram
P(<s> the man from Jupiter came </s>)
P(the|<s><s>) P(man|<s>the) P(from|the man) P(jupiter|man from) P(came|from jupiter) P(</s>|Jupiter came) P(</s>|came </s>)

N-Grams and Markov Models

The assumption that the probability of a word depends only on the previous word(s) is called **Markov assumption**.

Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far into the past.

•A **bigram** is called a **first-order** Markov model (because it looks one token into the past)
•A **trigram** is called a **second-order** Markov model;
•In general a **N-Gram** is called a **N-1 order** Markov model.

Estimating N-Gram Probabilities

Estimating n-gram probabilities is called **maximum likelihood estimation (**or **MLE**). We get the MLE estimate for the parameters of an n-gram model by ***getting counts from a corpus***, and **normalizing** the counts so that they lie between 0 and 1.

•**Estimating bigram probabilities:**

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

where C is the count of that pattern in the corpus

Estimating N-Gram probabilities

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

Estimating N-Gram Probabilities

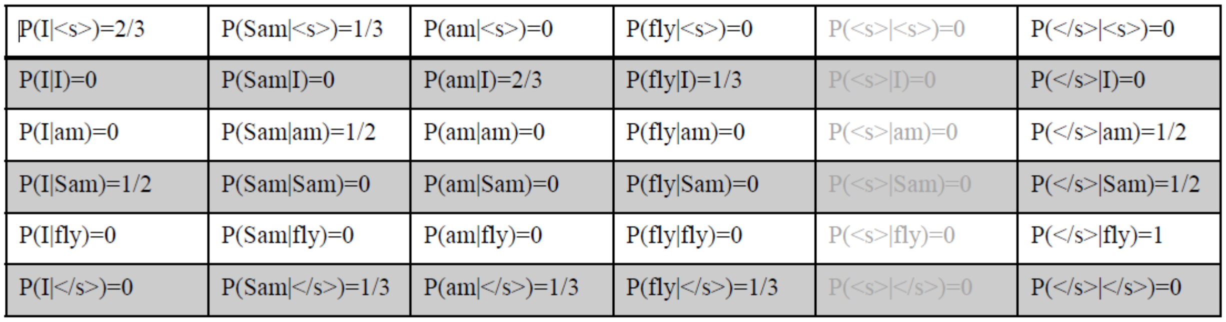
A Bigram Example

A mini-corpus: We augment each sentence with a special symbol <s> at the beginning of the sentence, to give us the bigram context of the first word, and special end-symbol </s>.

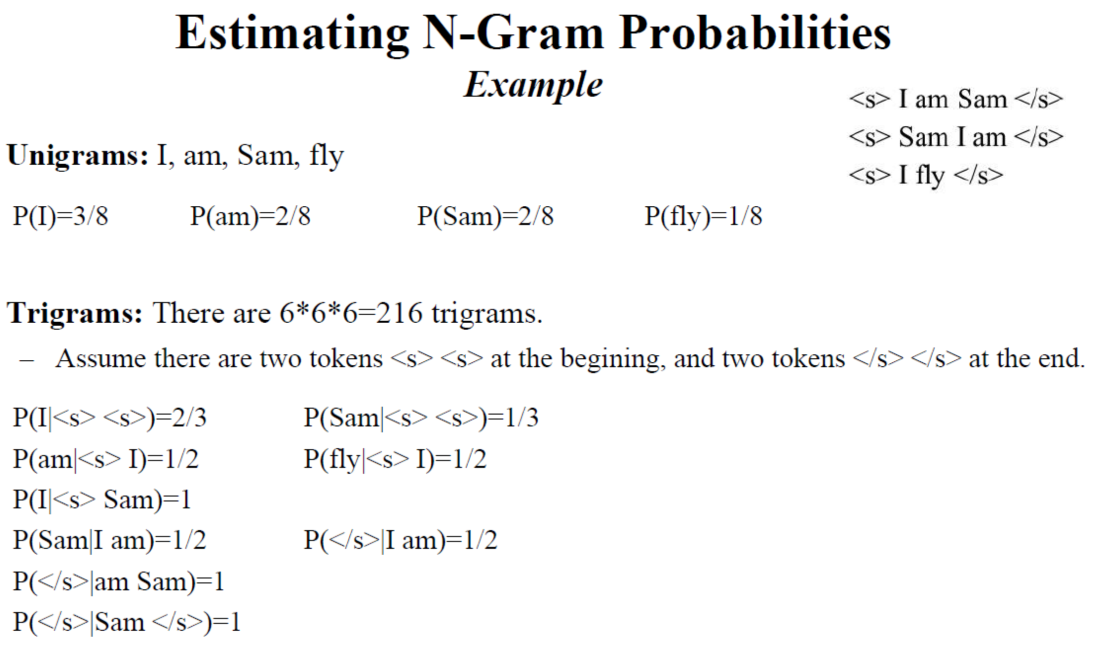
<s> I am Sam </s>
<s> Sam I am </s>
<s> I fly </s>

Unique words: I, am, Sam, fly

Bigrams: <s> and </s> are also tokens. There are 6(4+2) tokens and 6*6=36 bigrams



P(I|<s>)=2/3	P(Sam|<s>)=1/3	P(am|<s>)=0	P(fly|<s>)=0	P(<s>|<s>)=0	P(</s>|<s>)=0
P(I|I)=0	P(Sam|I)=0	P(am|I)=2/3	P(fly|I)=1/3	P(<s>|I)=0	P(</s>|I)=0
P(I|am)=0	P(Sam|am)=1/2	P(am|am)=0	P(fly|am)=0	P(<s>|am)=0	P(</s>|am)=1/2
P(I|Sam)=1/2	P(Sam|Sam)=0	P(am|Sam)=0	P(fly|Sam)=0	P(<s>|Sam)=0	P(</s>|Sam)=1/2
P(I|fly)=0	P(Sam|fly)=0	P(am|fly)=0	P(fly|fly)=0	P(<s>|fly)=0	P(</s>|fly)=1
P(I|</s>)=0	P(Sam|</s>)=1/3	P(am|</s>)=1/3	P(fly|</s>)=1/3	P(<s>|</s>)=0	P(</s>|</s>)=0



Estimating N-Gram Probabilities

Example

<s> I am Sam </s>
<s> Sam I am </s>
<s> I fly </s>

Unigrams: I, am, Sam, fly

P(I)=3/8	P(am)=2/8	P(Sam)=2/8	P(fly)=1/8

Trigrams: There are 6*6*6=216 trigrams.

– Assume there are two tokens <s> <s> at the begining, and two tokens </s> </s> at the end.

P(I|<s> <s>)=2/3	P(Sam|<s> <s>)=1/3
P(am|<s> I)=1/2	P(fly|<s> I)=1/2
P(I|<s> Sam)=1	
P(Sam|I am)=1/2	P(</s>|I am)=1/2
P(</s>|am Sam)=1	
P(</s>|Sam </s>)=1	

4.10 Language model Evaluation

Language model evaluation in Natural Language Processing (NLP) is a crucial step to assess the performance and quality of a language model. Evaluating language models helps researchers and developers understand how well a model performs on various tasks and benchmark it against other models or baselines. Here are some common evaluation techniques used in NLP:

1. **Perplexity:** Perplexity is a widely used metric for evaluating language models. It measures how well a language model predicts a given text corpus. Lower perplexity indicates better performance. Perplexity is calculated using the probabilities assigned by the model to each word in a test set.
2. **Accuracy:** Accuracy is a common evaluation metric for tasks like sentiment analysis, text classification, or named entity recognition. It measures the proportion of correctly predicted instances compared to the total number of instances in a test set. Accuracy is usually calculated by comparing the predicted labels or outputs of the model with the ground truth labels.
3. **F1 Score:** F1 score is a widely used metric for evaluating models in tasks like named entity recognition, part-of-speech tagging, and information extraction. It balances precision (the proportion of correctly predicted positive instances) and recall (the proportion of actual positive instances correctly predicted). F1 score is the harmonic mean of precision and recall and provides a single measure of model performance.
4. **BLEU Score:** BLEU (Bilingual Evaluation Understudy) is a metric commonly used in machine translation to evaluate the quality of generated translations. It measures the overlap between the model's predicted translations and reference translations. BLEU score ranges from 0 to 1, with higher scores indicating better translation quality.
5. **ROUGE Score:** ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a family of metrics used to evaluate text summarization systems. ROUGE measures the overlap between the model's generated summaries and reference summaries. Different variants of ROUGE, such as ROUGE-N (measuring n-gram overlap) and ROUGE-L (measuring longest common subsequence), are used to evaluate different aspects of summarization quality.
6. **Human Evaluation:** In addition to automated metrics, human evaluation is often conducted to assess the quality of language models. Human evaluators provide judgments on various aspects of model outputs, such as fluency, coherence, grammaticality, relevance, and overall quality. Human evaluation provides valuable insights that may not be captured by automated metrics.

It is important to note that the choice of evaluation metric depends on the specific NLP task and the goals of the evaluation. Different metrics have their strengths and weaknesses, and a combination of metrics is often used to provide a comprehensive evaluation of a language model's performance.

4.11 Parameter Estimation

Parameter estimation in natural language processing (NLP) refers to the process of estimating the values of model parameters based on observed data. In NLP, parameter estimation is crucial for various tasks, such as language modeling, machine translation, sentiment analysis, and part-of-speech tagging, among others.

Types of Parameter Estimations

1. Maximum-Likelihood Estimation and Smoothing (MLE)
2. Bayesian Parameter Estimation
3. Large-Scale Language Models

1.Maximum-Likelihood Estimation and Smoothing (MLE)

Maximum Likelihood Estimation (MLE) is a common technique used in Natural Language Processing (NLP) to estimate the parameters of a probabilistic language model. MLE aims to find the parameter values that maximize the likelihood of the observed data.

In the context of NLP, MLE is often used to estimate the probabilities of words or sequences of words in a language model. The basic idea is to count the occurrences of different words or sequences of words in a given training corpus and use these counts to calculate the probabilities as follows.

P(wi/wi-1,wi-2)=c(wi, wi-1, wi-2)/c(wi-1,wi-2)

Here's a step-by-step explanation of how MLE works in NLP:

1. **Data Collection:** Collect a large corpus of text that represents the domain or language you want to model. This corpus will be used as the training data.
2. **Data Preprocessing:** Preprocess the training data by tokenizing it into words or subword units, removing punctuation, normalizing case, etc. This step ensures that the data is in a suitable format for modeling.
3. **Parameter Estimation:** Calculate the probabilities of words or sequences of words based on the training data. For example, to estimate the probability of a word, count the number of occurrences of that word in the corpus and divide it by the total number of words in the corpus.
4. **Smoothing:** One challenge in using MLE is that unseen words or sequences in the training data will have zero probabilities. To address this issue, smoothing techniques are often applied to assign non-zero probabilities to unseen events. Common smoothing methods include Laplace smoothing (add-one smoothing), Lidstone smoothing, and Good-Turing smoothing.

5. **Model Evaluation:** Evaluate the performance of the language model using evaluation metrics such as perplexity, accuracy, or F1 score. These metrics help assess how well the model predicts the target language or performs on specific NLP tasks.

MLE is a fundamental approach for estimating language model probabilities in NLP. However, it has some limitations. MLE assumes that the training data is representative of the target distribution and that each data point is independent. In practice, these assumptions may not always hold, leading to potential limitations in the performance of the language model. Researchers and practitioners often explore more sophisticated techniques, such as neural language models, to overcome these limitations and improve language modeling in NLP.

2. Bayesian Parameter Estimation

Bayesian parameter estimation is an alternative approach to estimating the parameters of a language model in Natural Language Processing (NLP). Unlike Maximum Likelihood Estimation (MLE), which provides point estimates of the parameters, Bayesian estimation incorporates prior knowledge and uncertainty into the parameter estimation process.

In Bayesian parameter estimation, the goal is to estimate the posterior distribution of the parameters given the observed data and prior knowledge. The posterior distribution represents the updated belief about the parameter values after considering the observed data. This distribution is obtained by combining the likelihood of the data and the prior distribution of the parameters using Bayes' theorem.

Here's a step-by-step explanation of how Bayesian parameter estimation works in NLP:

1. Define a Prior Distribution: Specify a prior distribution that captures the initial belief or knowledge about the parameters before observing any data. The prior distribution represents the uncertainty or prior information about the parameter values.
2. Likelihood Calculation: Calculate the likelihood of the observed data given the parameters. The likelihood measures how probable the observed data is under the current parameter values.
3. Bayesian Inference: Apply Bayes' theorem to update the prior distribution based on the likelihood and obtain the posterior distribution of the parameters. The posterior distribution reflects the updated belief about the parameter values after considering the observed data.
4. Posterior Analysis: Analyze the posterior distribution to obtain estimates of the parameters. This can be done by calculating the mean, median, or mode of the posterior distribution, which represent the point estimates of the parameters. Additionally, credible intervals can be computed to quantify the uncertainty in the parameter estimates.
5. Model Evaluation: Evaluate the performance of the Bayesian language model using evaluation metrics such as perplexity, accuracy, or F1 score, similar to the evaluation of MLE-based models.

Bayesian parameter estimation provides several advantages over MLE in NLP. It allows the incorporation of prior knowledge, which can be particularly useful when the amount of available training data is limited. Additionally, Bayesian estimation provides a richer representation of uncertainty by yielding a posterior distribution rather than a point estimate. This uncertainty information can be valuable for decision-making or downstream applications.

However, Bayesian parameter estimation can be computationally demanding, especially for complex models with high-dimensional parameter spaces. Techniques like Markov Chain Monte Carlo (MCMC) or variational inference are commonly used to approximate the posterior distribution when direct computation is infeasible.

Overall, Bayesian parameter estimation offers a principled and flexible framework for estimating parameters in language models, incorporating prior knowledge, and capturing uncertainty in NLP tasks.

4.12 Language Model Adaption

Language model adaptation in NLP refers to the process of fine-tuning a pre-trained language model on a specific task or domain to improve its performance. Language models like GPT-3.5 are trained on a large corpus of diverse text from the internet, which gives them a general understanding of language. However, fine-tuning or adapting them to a specific task or domain can enhance their performance on that particular task.

Here are the general steps involved in language model adaptation:

1. **Dataset Collection:** Gather a dataset that is specific to the target task or domain you want to adapt the language model to. This dataset should be representative of the target task and should include text samples that the language model is likely to encounter during inference.
2. **Dataset Preprocessing:** Preprocess the collected dataset by cleaning the text, removing irrelevant information, and ensuring the dataset is in a format that can be used for training.
3. **Model Architecture:** Decide on the specific architecture you want to use for adaptation. This can be the same architecture as the pre-trained language model or a modified version depending on the task requirements.
4. **Task-Specific Labels:** If your target task involves supervised learning, you will need to annotate or label your dataset with the appropriate task-specific labels. This step is crucial for tasks like sentiment analysis, named entity recognition, or machine translation.
5. **Fine-tuning:** Initialize the pre-trained language model with the weights learned during pre-training, and then fine-tune it on your target dataset. During fine-tuning, the model learns task-specific patterns and representations. The process typically involves minimizing a task-specific loss function using techniques such as backpropagation and gradient descent.
6. **Hyperparameter Tuning:** Experiment with different hyperparameters such as learning rate, batch size, and regularization techniques to optimize the performance of the adapted model. This step helps to find the best set of hyperparameters that work well for the specific task.
7. **Evaluation and Iteration:** Evaluate the performance of the adapted language model on a validation set or using other evaluation metrics specific to your task. Iterate on the fine-tuning

process by making changes to the architecture, hyperparameters, or dataset, if necessary, to further improve performance.

It's worth noting that language model adaptation requires a substantial amount of task-specific data to achieve good performance. If only a limited amount of data is available, transfer learning techniques such as few-shot or zero-shot learning can be used to leverage the knowledge learned from the pre-trained language model. Additionally, it's important to strike a balance between task-specific adaptation and preserving the general language understanding of the original pre-trained model.

4.13 Types of Language Models

There are various types of language models used in natural language processing (NLP) that differ in their architecture, training methods, and intended use cases. Here are some commonly used types of language models in NLP:

1. **N-gram Language Models**: N-gram models are simple and widely used language models that predict the probability of a word given its preceding context of n-1 words. They rely on counting and statistical techniques to estimate the probabilities. For example, a trigram model predicts the next word based on the previous two words.
2. **Neural Language Models**: Neural language models leverage deep learning techniques, such as recurrent neural networks (RNNs), long short-term memory (LSTM) networks, and transformers, to capture complex patterns and dependencies in language. These models learn distributed representations of words and generate probabilities based on the context.
3. **Transformer-based Language Models**: Transformer models, such as OpenAI's GPT (Generative Pre-trained Transformer), have gained significant attention in recent years. These models use self-attention mechanisms to capture global dependencies and parallelize computation, making them highly effective for tasks like language generation, machine translation, and question answering.
4. **Contextualized Language Models**: Contextualized language models, such as ELMo (Embeddings from Language Models) and BERT (Bidirectional Encoder Representations from Transformers), generate word representations that are sensitive to the context in which the word appears. These models provide contextual embeddings that are useful for downstream NLP tasks like sentiment analysis, named entity recognition, and text classification.
5. **Encoder-Decoder Models**: Encoder-decoder models, often based on the sequence-to-sequence architecture, are used for tasks like machine translation and text summarization. These models consist of an encoder that processes the input sequence and a decoder that generates the output sequence based on the encoded representation.
6. **Hybrid Models**: Some language models combine multiple approaches or architectures to leverage their individual strengths. For example, ULMFiT (Universal Language Model Fine-tuning) combines features of transformer models with techniques like transfer learning and fine-tuning to improve performance on specific tasks.
7. **Pre-trained Language Models**: Pre-trained language models, such as GPT, BERT, and RoBERTa, are models that are trained on large amounts of data before being fine-tuned for specific tasks. These models capture a broad understanding of language and can be adapted to various downstream tasks with additional training.

These are just a few examples of the types of language models used in NLP. Different models have different strengths and are suited for specific tasks and scenarios. Researchers continue to explore new architectures and techniques to develop more powerful and efficient language models for a wide range of NLP applications.

4.14 Language-Specific Modelling Problems

In natural language processing (NLP), there are several language-specific modeling problems that arise due to the unique characteristics and complexities of different languages. Here are some common language-specific modeling challenges in NLP:

1. **Morphological Variations:** Many languages exhibit rich morphology, where words can have multiple forms depending on factors like tense, number, gender, and case. Modeling morphological variations can be challenging, especially for languages with highly inflected forms, as it requires capturing the intricate relationships between word forms and their meanings.
2. **Word Order and Syntax:** Languages vary in their word order and sentence structure. For example, English follows a subject-verb-object (SVO) order, while languages like Japanese follow a subject-object-verb (SOV) order. Modeling the syntactic structure and understanding the correct word order for different languages is crucial for tasks like parsing, machine translation, and text generation.
3. **Named Entity Recognition (NER):** Named Entity Recognition involves identifying and classifying named entities such as names of people, organizations, locations, and dates in text. Different languages have specific naming conventions and variations, which pose challenges for building language-independent NER systems. Language-specific rules and resources are often required to handle these variations effectively.
4. **Sentiment Analysis:** Sentiment analysis aims to determine the sentiment or opinion expressed in a text. The sentiment lexicons, patterns, and linguistic cues that indicate sentiment can differ across languages. Building accurate sentiment analysis models requires language-specific sentiment resources and training data that capture the nuances of sentiment expression in different languages.
5. **Language Idiosyncrasies:** Each language has its own set of idiosyncrasies, such as idiomatic expressions, proverbs, and cultural references. These language-specific nuances pose challenges for models that aim to understand and generate natural language text. Capturing and representing these idiosyncrasies is crucial for building language models that accurately handle language-specific contexts.
6. **Low-Resource Languages:** Low-resource languages, which have limited amounts of training data and linguistic resources, present unique modeling challenges. Limited resources make it difficult to develop robust models for these languages. Techniques like cross-lingual transfer learning, data augmentation, and unsupervised methods are often employed to address the scarcity of language-specific resources.
7. **Code-Switching and Multilingualism:** Many languages exhibit code-switching, where speakers alternate between two or more languages within a conversation. Modeling code-switching and multilingual text requires techniques that can handle language mixing, language identification, and understanding the context-dependent language usage.

Addressing these language-specific modeling challenges in NLP often involves a combination of linguistic knowledge, domain expertise, and language-specific resources. Researchers and practitioners work on developing techniques that can effectively handle these challenges and improve the performance and applicability of NLP models across different languages.

4.15 Multilingual and Cross lingual Language Modelling.

Multilingual and cross-lingual language modeling in NLP involves developing models that can understand, generate, or process multiple languages. Here's an overview of multilingual and cross-lingual language modeling:

1. **Multilingual Language Modeling:** Multilingual language models are trained to handle multiple languages simultaneously. These models are typically trained on a diverse multilingual corpus and aim to capture shared linguistic properties across languages. By leveraging the similarities between languages, multilingual models can generalize well across different language contexts. They can be used for tasks like text classification, named entity recognition, and sentiment analysis in multiple languages.
2. **Cross-lingual Language Modeling:** Cross-lingual language models focus on bridging the gap between different languages. These models are trained on a source language and then applied to a target language, even if the target language has little or no labeled data. Cross-lingual models enable transfer learning, where knowledge learned from a resource-rich language can be transferred to improve performance in a low-resource language. This approach is particularly useful for tasks like machine translation, cross-lingual document retrieval, and cross-lingual information extraction.
3. **Bilingual and Multilingual Embeddings:** Bilingual and multilingual word embeddings represent words from different languages in a shared vector space. These embeddings capture semantic and syntactic similarities between words in multiple languages. Bilingual embeddings map words from one language to their counterparts in another language, while multilingual embeddings aim to represent words from multiple languages in a single vector space. These embeddings facilitate cross-lingual tasks by allowing for direct comparisons and knowledge transfer across languages.
4. **Machine Translation:** Machine translation is a classic example of cross-lingual language modeling. Neural machine translation models, such as sequence-to-sequence models, use an encoder-decoder architecture to translate text from one language to another. These models can be trained with parallel corpora containing source-target language pairs, and they learn to map the source language to the target language. Transfer learning techniques and multilingual training can improve the translation performance across multiple languages.
5. **Zero-Shot and Few-Shot Learning:** Zero-shot and few-shot learning techniques enable cross-lingual transfer without requiring labeled data in the target language. In zero-shot

learning, a model can generate outputs in a language it has not been explicitly trained on, using only high-level instructions or prompts. Few-shot learning involves training a model on a limited amount of labeled data from the target language, allowing it to perform well on that language. These techniques are useful when resources in the target language are scarce.

6. **Cross-lingual Named Entity Recognition (NER):** Cross-lingual NER involves recognizing named entities in multiple languages. By leveraging multilingual training data and shared representations, cross-lingual NER models can generalize across languages and recognize entities even in low-resource languages. Techniques like cross-lingual transfer learning and alignment methods are used to improve cross-lingual NER performance.

Multilingual and cross-lingual language modeling techniques have contributed significantly to making NLP more accessible and effective across different languages and language pairs. They enable the development of models that can handle diverse linguistic contexts, promote knowledge transfer, and facilitate communication and understanding in a multilingual world.