WHAT IS NLP?

- Natural Language Processing (NLP) allows machines to break down and interpret human language. It's at the core of tools we use every day – from translation software, chatbots, spam filters, and search engines, to grammar correction software, voice assistants, and social media monitoring tools.

WHAT IS NLP?

- NLP makes it possible for computers to understand the human language. Behind the scenes, NLP analyzes the grammatical structure of sentences and the individual meaning of words, then uses algorithms to extract meaning and deliver outputs. In other words, it makes sense of human language so that it can automatically perform different tasks.

- Probably, the most popular examples of NLP in action are virtual assistants, like Google Assist, Siri, and Alexa. NLP understands written and spoken text like *"Hey Siri, where is the nearest gas station?"* and transforms it into numbers, making it easy for machines to understand.

- Another well-known application of NLP is chatbots. They help support teams solve issues by understanding common language requests and responding automatically.

- There are many other everyday apps you use, where you've probably encountered NLP without even noticing. Text recommendations when writing an email, offering to translate a Facebook post written in a different language, or filtering unwanted promotional emails into your spam folder.
- In a nutshell, the goal of Natural Language Processing is to make human language – which is complex, ambiguous, and extremely diverse – easy for machines to understand.

DIFFERENCES BETWEEN AI, ML AND NLP

- AI is an umbrella term for machines that can simulate human intelligence. AI encompasses systems that mimic cognitive capabilities, like learning from examples and solving problems. This covers a wide range of applications, from self-driving cars to predictive systems. Both ML and NLP are subsets of AI

- Natural Language Processing (NLP) deals with how computers understand and translate human language. With NLP, machines can make sense of written or spoken text and perform tasks like translation, keyword extraction, topic classification, and more.

- But to automate these processes and deliver accurate responses, you'll need machine learning. Machine learning is the process of applying algorithms that teach machines how to automatically learn and improve from experience without being explicitly programmed.

- AI-powered chatbots, for example, use NLP to interpret what users say and what they intend to do, and machine learning to automatically deliver more accurate responses by learning from past interactions.

COMPONENTS OF NLP

There are two main components of NLP -

1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

- NLU mainly used in Business applications to understand the customer's problem in both spoken and written language. NLU involves the following tasks -
- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

- Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

NLP TASKS

Natural language processing tasks involve syntactic and semantic analysis, used to break down human language into machine-readable chunks.

- **Syntactic analysis**, also known as parsing or syntax analysis, identifies the syntactic structure of a text and the dependency relationships between words, represented on a diagram called a parse tree.
- **Semantic analysis** focuses on identifying the meaning of language. However, since language is polysemic and ambiguous, semantics is considered one of the most challenging areas in NLP.
- Semantic tasks analyze the structure of sentences, word interactions, and related concepts, in an attempt to discover the meaning of words, as well as understand the topic of a text.

Some of the subtasks of Syntax and Semantic Analysis are:

1. Tokenization

- Tokenization is an essential task in natural language processing used to break up a string of words into semantically useful units called *tokens*.
- Sentence tokenization splits sentences within a text, and word tokenization splits words within a sentence. Generally, word tokens are separated by blank spaces, and sentence tokens by stops. However, you can perform high-level tokenization for more complex structures, like words that often go together, otherwise known as collocations (e.g., *New York*).
- Here's an example of how word tokenization simplifies text:
- Customer service couldn't be better! = "customer service" "could" "not" "be" "better".

2. **Part-of-speech tagging**

- Part-of-speech tagging (abbreviated as PoS tagging) involves adding a part of speech category to each token within a text. Some common PoS tags are *verb*, *adjective*, *noun*, *pronoun*, *conjunction*, *preposition*, *intersection*, among others. In this case, the example above would look like this:
- *"Customer service": NOUN, "could": VERB, "not": ADVERB, be": VERB, "better": ADJECTIVE, "!": PUNCTUATION*
- PoS tagging is useful for identifying relationships between words and, therefore, understand the meaning of sentences.

3.PARSING: What is parsing? According to the dictionary, to parse is to "resolve a sentence into its component parts and describe their syntactic roles."
That actually nailed it but it could be a little more comprehensive. Parsing refers to the formal analysis of a sentence by a computer into its constituents, which results in a parse tree showing their syntactic relation to one another in visual form, which can be used for further processing and understanding.

A parse tree is a tree that highlights the syntactical structure of a sentence according to a formal grammar, for example by exposing the relationships between words or sub-phrases. Depending on which type of grammar we use, the resulting tree will have different features.

You can try different parsing algorithms and strategies depending on the nature of the text you intend to analyze, and the level of complexity you'd like to achieve.Parsing may be top down or bottom up.
Since they are based on totally different assumptions, the resulting trees will be very different. Although, in both cases, the end goal is to extract syntactic information.We will xplore examples of 2 Parsing techniques :

CONSTITUENCY PARSING

- Below is a parse tree for the sentence "The thief robbed the apartment." Included is a description of the three different information types conveyed by the sentence.

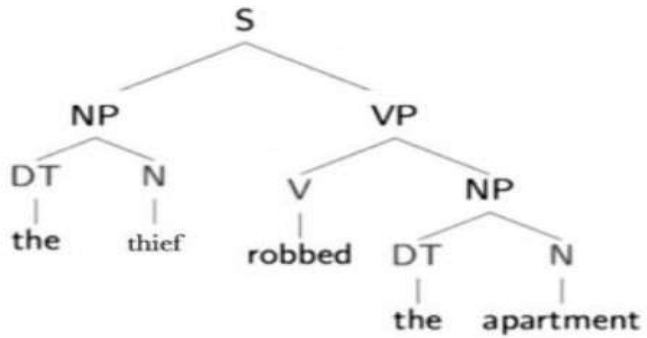
1) Part of speech

N = noun
V = verb
DT = determiner

2) Phrases

Noun Phrases: : "the thief", "the apartment"
Verb Phrases: "robbed the apartment"
Sentence: "the burglar robbed the apartment"

3) Relationships



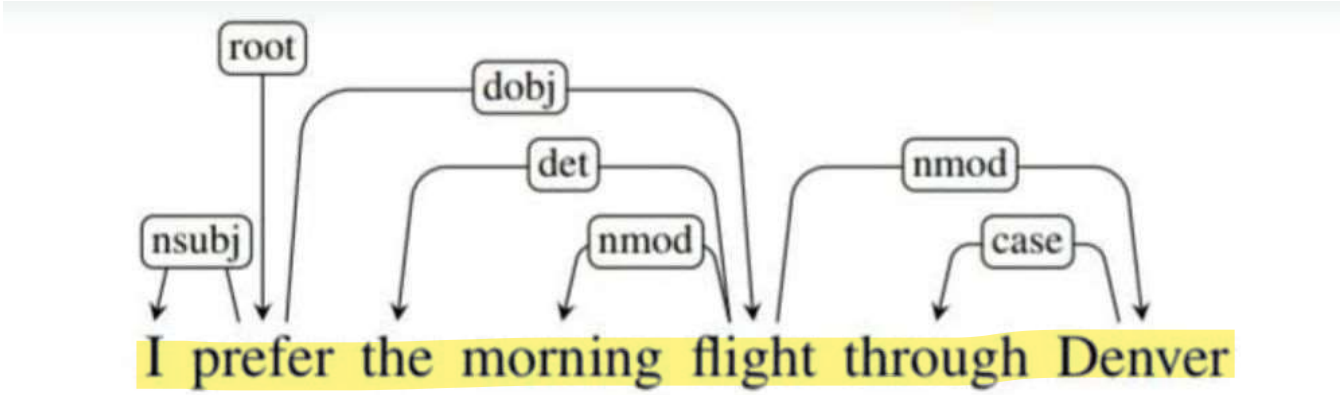
The letters directly above the single words show the parts of speech for each word (noun, verb and determiner). One level higher is some hierarchical grouping of words into phrases. For example, "the thief" is a noun phrase, "robbed the apartment" is a verb phrase and when put together the two phrases form a sentence, which is marked one level higher.

But what is actually meant by a noun or verb phrase? Noun phrases are one or more words that contain a noun and maybe some descriptors, verbs or adverbs. The idea is to group nouns with words that are in relation to them.
A parse tree also provides us with information about the grammatical relationships of the words due to the structure of their representation. For example, we can see in the structure that "the thief" is the subject of "robbed.“

With structure I mean that we have the verb ("robbed"), which is marked with a "V" above it and a "VP" above that, which is linked with a "S" to the subject ("the thief"), which has a "NP" above it. This is like a template for a subject-verb relationship and there are many others for other types of relationships.

DEPENDENCY PARSING

- **Dependency parsing**
- Dependency parsing is the task of extracting a dependency parse of a sentence that represents its grammatical structure and defines the relationships between “head” words and words, which modify those heads.
- Relations among the words are illustrated above the sentence with directed, labeled arcs from heads to dependents (+ indicates the dependent).



4.Stemming: Stemming is used to normalize words into its base form or root form.

For example, celebrates, celebrated and celebrating, all these words are originated with a single root word "celebrate." The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

- Search engines use stemming for indexing the words. That's why rather than storing all forms of a word, a search engine can store only the stems. In this way, stemming reduces the size of the index and increases retrieval accuracy. This indiscriminate cutting can be successful in some occasions, but not always, and that is why we affirm that this approach presents some limitations. The big problem with stemming is that sometimes it produces the root word which may not have any meaning.
- **For Example,** intelligence, intelligent, and intelligently, all these words are originated with a single root word "intelligen." In English, the word "intelligen" do not have any meaning.

5. Lemmatization : Lemmatization is quite similar to the Stemming. It is used to group different inflected forms of the word, called Lemma. The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

- **For example:** In lemmatization, the words intelligence, intelligent, and intelligently has a root word intelligent, which has a meaning.
- While lemmatization is dictionary-based and chooses the appropriate lemma based on context, stemming operates on single words without considering the context. For example, in the sentence:
- *"This is better"*
- The word "better" is transformed into the word "good" by a lemmatizer but is unchanged by stemming.

6. Stopword Removal

- Removing stop words is an essential step in NLP text processing. It involves filtering out high-frequency words that add little or no semantic value to a sentence, for example, *which, to, at, for, is,* etc.
- You can even customize lists of stopwords to include words that you want to ignore.
- Let's say you want to classify customer service tickets based on their topics. In this example: *"Hello, I'm having trouble logging in with my new password"*, it may be useful to remove stop words like ***"hello"***, ***"I"***, ***"am"***, ***"with"***, ***"my"***, so you're left with the words that help you understand the topic of the ticket: ***"trouble"***, ***"logging in"***, ***"new"***, ***"password"***.

7. Named entity recognition (NER) concentrates on determining which items in a text (i.e. the "named entities") can be located and classified into pre-defined categories. These categories can range from the names of persons, organizations and locations to monetary values and percentages.

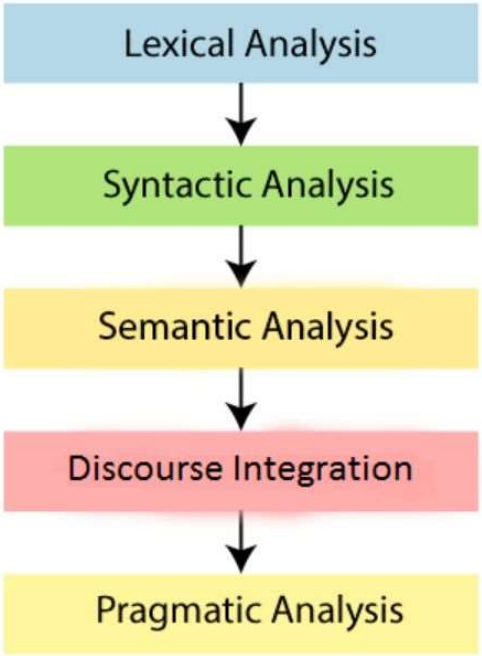
- For example:
- Before NER: *Martin bought 300 shares of SAP in 2016.*
- After NER: *[Martin]Person bought 300 shares of [SAP]Organization in [2016]Time.*

8. **Relationship Extraction** takes the named entities of NER and tries to identify the semantic relationships between them. For example, in the phrase *"Susan lives in Los Angeles,"* a person *(Susan)* is related to a place *(Los Angeles)* by the semantic category "lives in."

- Text Classification
- Text classification is the process of understanding the meaning of unstructured text and organizing it into predefined categories (tags). One of the most popular text classification tasks is sentiment analysis, which aims to categorize unstructured data by sentiment.
- With the use of sentiment analysis, for example, we may want to predict a customer's opinion and attitude about a product based on a review they wrote. Sentiment analysis is widely applied to reviews, surveys, documents and much more.These can be categorized as positive, negative or neutral.

PHASES OF NLP

- There are 5 main phases of NLP



PHASES OF NLP

1.Lexical Analysis and Morphological

- The first phase of NLP is the Lexical Analysis.It is also known as Morphological Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexions or morphemes. It divides the whole text into paragraphs, sentences, and words.The most commonly used Lexicon Normalization Techniques are Stemming and Lemmatization.

2. Syntactic Analysis (Parsing)

- Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

3. Semantic Analysis

- Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences. It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as "hot ice-cream". During semantic analysis two main operations are executed
- First, each separate word will be mapped with appropriate objects in the database. The dictionary meaning of every word will be found. A word might have more than one meaning.
- Secondly, all the meanings of each different word will be integrated to find a proper correlation between the word structures. This process of determining the correct meaning is called lexical disambiguation. It is done by associating each word with the context.
- This process defined above can be used to determine the partial meaning of a sentence. However semantic and syntax are two completely contrasting concepts. It might be possible that a syntactically correct sentence is semantically incorrect. For example, "**A rock smelled the colour nine**." It is syntactically correct as it obeys all the rules of English, but is semantically incorrect.

4. DISCOURSE INTEGRATION

- Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.
- Discourse deals with the effect of a previous sentence on the sentence in consideration. In the text, “Jack is a bright student. He spends most of the time in the library.” Here, discourse assigns “he” to refer to “Jack”.

5. PRAGMATIC ANALYSIS

The final stage of NLP, Pragmatics interprets the given text using information from the previous steps. Given a sentence, “Turn off the lights” is an order or request to switch off the lights.

- Pragmatics focuses on the **effects of context on meaning**, and Discourse Analysis studies written and spoken language in relation to its social context. pragmatics has more to do with the external or physical context, while discourse analysis focuses on the linguistic context.

Why is NLP Hard?

Ambiguity, generally used in natural language processing, can be referred as the ability of being understood in more than one way. In simple terms, we can say that ambiguity is the capability of being understood in more than one way. Natural language is very ambiguous. NLP has the following types of ambiguities −

Lexical Ambiguity

- The ambiguity of a single word is called lexical ambiguity. For example, treating the word **silver** as a noun, an adjective, or a verb.

Syntactic Ambiguity

- This kind of ambiguity occurs when a sentence is parsed in different ways. For example, the sentence “The man saw the girl with the telescope”. It is ambiguous whether the man saw the girl carrying a telescope or he saw her through his telescope

.

Semantic Ambiguity

- This kind of ambiguity occurs when the meaning of the words themselves can be misinterpreted. In other words, semantic ambiguity happens when a sentence contains an ambiguous word or phrase. For example, the sentence “The car hit the pole while it was moving” is having semantic ambiguity because the interpretations can be “The car, while moving, hit the pole” and “The car hit the pole while the pole was moving”.

Anaphoric Ambiguity

- Anaphoric means being a word or phrase that takes its reference from another word or phrase and especially from a preceding word or phrase
- This kind of ambiguity arises due to the use of anaphora entities in discourse. For example, the horse ran up the hill. It was very steep. It soon got tired. Here, the anaphoric reference of “it” in two situations cause ambiguity.

Pragmatic ambiguity

- Such kind of ambiguity refers to the situation where the context of a phrase gives it multiple interpretations. In simple words, we can say that pragmatic ambiguity arises when the statement is not specific. For example, the sentence “I like you too” can have multiple interpretations like I like you (just like you like me), I like you (just like someone else dose).

- Syntactic Error

little has a Mary Lamb

Semantic Error

- ? Colourless green ideas sleep furiously

Why is NLP hard?

There are probably 4 main things that we pay attention to in any normal face to face conversation

1. **Physical gestures** — the expressions on their face and the gestures that they make with their hand as they explain some extravagant story they've been through.

2. **Inflections** — *how* they say the words they are saying. Are they raising their voice to show anger? Are they trying to do an impersonation of someone that they're ridiculing?

3. **Content** — *what* they are explicitly saying, the words that are explicitly being spoken.

4. **Context** — is what is being said relevant to what you are currently talking about? Or is it a quick diversion, referencing something else?

- But now replay that conversation you had but as if you were speaking over the phone. You're now relying on 2/3 of those inputs you had, the inflection of their voice and the content, probably resulting in some uncertainty at certain parts of the conversation.
- Now,if you were texting each other, you are trying to understand the person operating on as little information as possible

ambiguity
context and sentiments associated with the content

Classical Problems in NLP

NLP is a powerful tool with huge benefits, but there are still a number of Natural Language Processing limitations and problems:

Contextual words and phrases and homonyms

- The same words and phrases can have different meanings according the context of a sentence and many words – especially in English – have the exact same pronunciation but totally different meanings.
- For example:
- *I **ran** to the store because we **ran** out of milk.*
- *Can I **run** something past you real quick?*
- *The house is looking really **run** down.*
- These are easy for humans to understand because we read the context of the sentence and we understand all of the different definitions. And, while NLP language models may have learned all of the definitions, differentiating between them in context can present problems.
- **Homonyms** – two or more words that are pronounced the same but have different definitions – can be problematic for question answering and speech-to-text applications because they aren't written in text form. Usage of *their* and *there*, for example, is even a common problem for humans.

ice ball

i see ball

Classical Problems in NLP

Synonyms

Synonyms can lead to issues similar to contextual understanding because we use many different words to express the same idea. So, for building NLP systems, it's important to include all of a word's possible meanings and all possible synonyms. Text analysis models may still occasionally make mistakes, but the more relevant training data they receive, the better they will be able to understand synonyms.

Irony and sarcasm

Irony and sarcasm present problems for machine learning models because they generally use words and phrases that, strictly by definition, may be positive or negative, but actually connote the opposite.

Ambiguity: Already discussed

Errors in text and speech

Misspelled or misused words can create problems for text analysis. Autocorrect and grammar correction applications can handle common mistakes, but don't always understand the writer's intention.

With spoken language, mispronunciations, different accents, stutters, etc., can be difficult for a machine to understand. However, as language databases grow and smart assistants are trained by their individual users, these issues can be minimized.

Classical Problems in NLP

Colloquialisms and slang:

- Informal phrases, expressions, idioms, and culture-specific lingo present a number of problems for NLP – especially for models intended for broad use. Because as formal language, colloquialisms may have no "dictionary definition" at all, and these expressions may even have different meanings in different geographic areas. Furthermore, cultural slang is constantly morphing and expanding, so new words pop up every day.
- This is where training and regularly updating custom models can be helpful, although it oftentimes requires quite a lot of data.

poppin'

Slangs in English are the words used in specific meanings instead of original meanings.
like grass is used instead of marijuana

Domain-specific language:

- Different businesses and industries often use very different language. An NLP processing model needed for healthcare, for example, would be very different than one used to process legal documents.

- A **morpheme** is the smallest meaningful lexical item in a language. A morpheme is not a word. The difference between a morpheme and a word is that a morpheme sometimes does not stand alone, but a word on this definition always stands alone. The field of linguistic study dedicated to morphemes is called morphology.
- Morphological parsing, in natural language processing, is **the process of determining the morphemes from which a given word is constructed**. ... The generally accepted approach to morphological parsing is through the use of a finite state transducer (FST),

Examples: train, fat

- er (comparative morpheme)
- S(plural morpheme)

FST can be used to find the morphemes in the sentence.

Affixes may be in form of suffixes or prefixes

Identify no. of morphemes

Judgers?? Blackened??

Free and Bound Morphemes

Free are standalone morphemes which have independent meaning eg: grace

Bound morphemes have meanings when used with other morphemes

How many different words are there?

Inflection creates different forms of the same word:

Verbs: to be, being, I am, you are, he is, I was,

Nouns: one book, two books

boy boys

Derivation creates different words from the same lemma:

grace ⇒ disgrace ⇒ disgraceful ⇒ disgracefully

boy boyish (noun adjective)

Compounding combines two words into a new word:

cream ⇒ ice cream ⇒ ice cream cone ⇒ ice cream cone bakery

Word formation is productive: New words are subject to all of these processes:

Morphemes, Stems & Affixes

dis-grace-ful-ly
prefix-stem-suffix-suffix

Many word forms consist of a stem plus a number of affixes (*prefixes* or *suffixes*)

Infixes are inserted inside the stem.

Circumfixes (German gesehen) surround the stem

Morphemes: the smallest (meaningful/grammatical) parts of words.

Stems (grace) are often free morphemes.

Free morphemes can occur by themselves as words.

Affixes (dis-, -ful, -ly) are usually bound morphemes.

Finite State Automaton

A finite-state automaton $M = \langle Q, \Sigma, q_0, F, \delta \rangle$ consists of:

- A finite set of states $Q = \{q_0, q_1, .., q_n\}$
- A finite alphabet Σ of input symbols (e.g. $\Sigma = \{a, b, c, ...\}$)
- A designated start state $q_0 \in Q$
- A set of final states $F \subseteq Q$
- A transition function δ:
  - The transition function for a deterministic (D)FSA: $Q \times \Sigma \rightarrow Q$

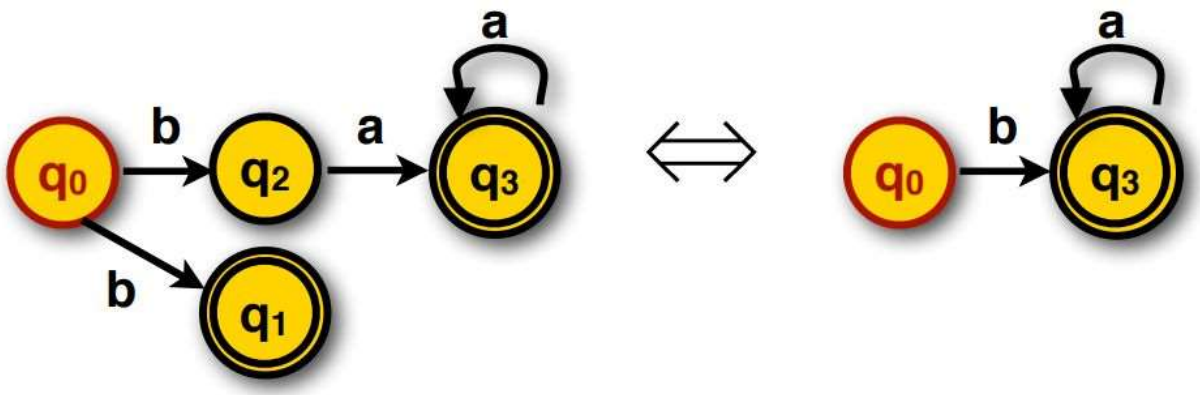
    $\delta(q,w) = q'$ for $q, q' \in Q, w \in \Sigma$

    If the current state is q and the current input is w, go to q'
  - The transition function for a nondeterministic (N)FSA: $Q \times \Sigma \rightarrow 2^Q$

    $\delta(q,w) = Q'$ for $q \in Q, Q' \subseteq Q, w \in \Sigma$

    If the current state is q and the current input is w, go to any $q' \in Q'$

Every NFA can be transformed into an equivalent DFA:



Recognition of a string *w* with a DFA is linear in the length of *w*

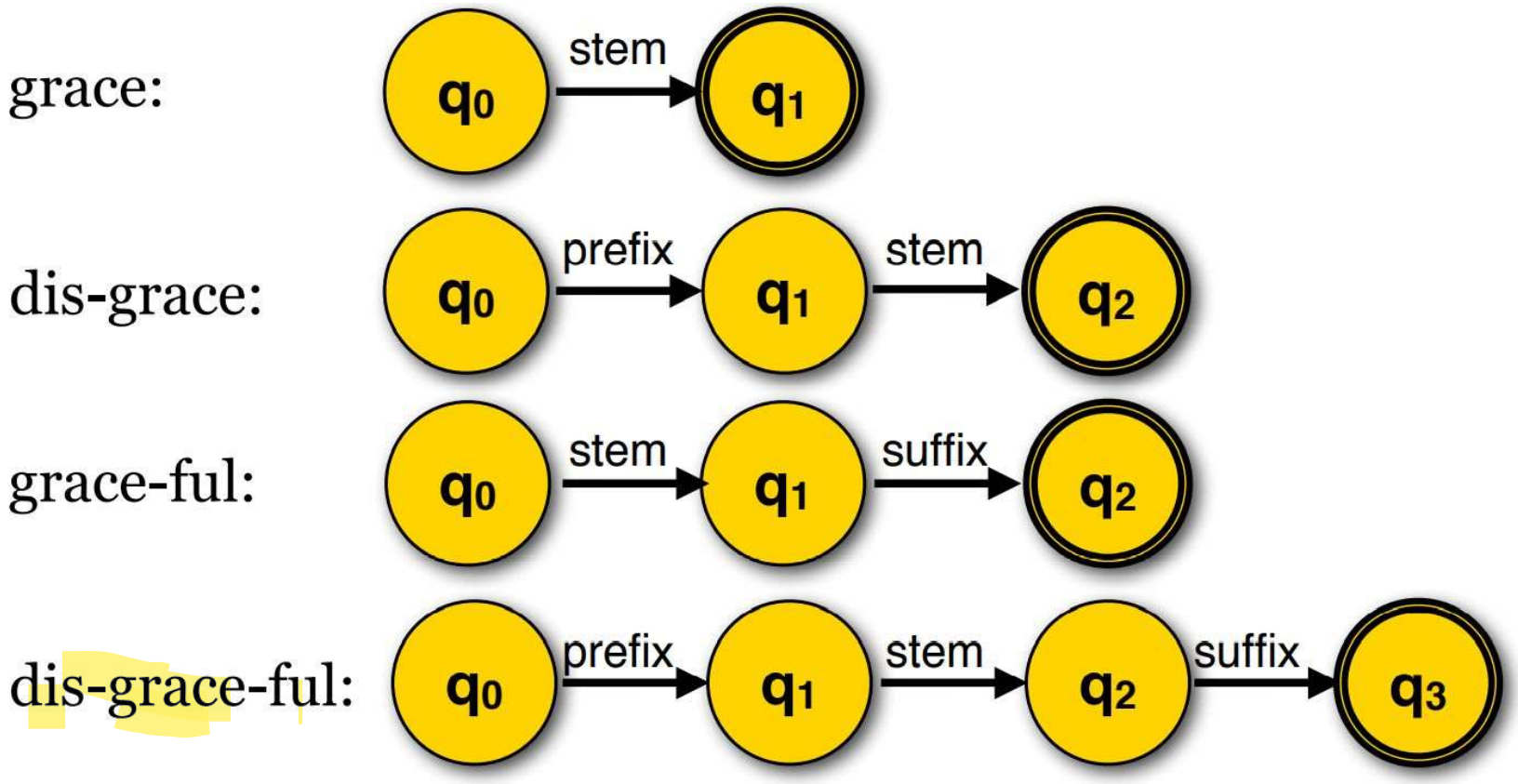
Finite-state automata define the class of regular languages

$L_1 = \{ a^n b^m \} = \{ab, aab, abb, aaab, abb, \ldots\}$ is a regular language,

$L_2 = \{ a^n b^n \} = \{ab, aabb, aaabbb, \ldots\}$ is not (it's context-free).

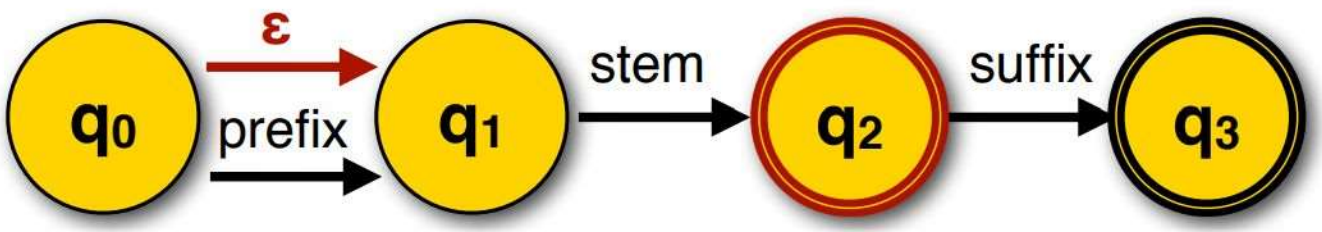
You cannot construct an FSA that accepts all the strings in L_2 and nothing else.

Finite State Automaton for Morphology

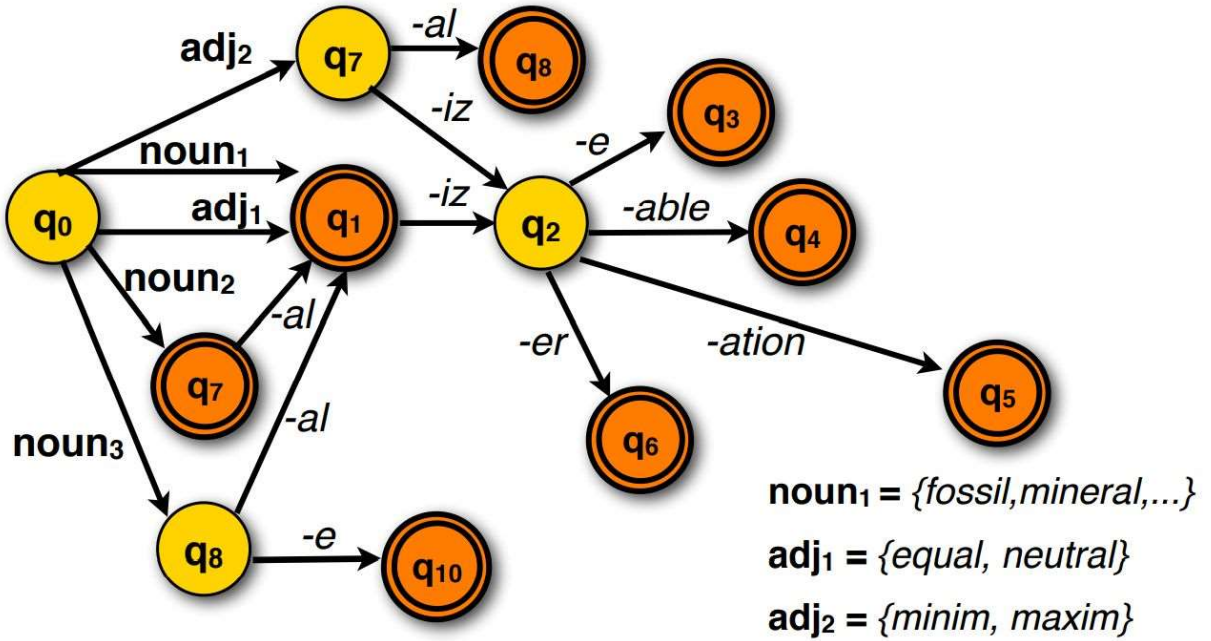


Union: Merging Automaton

grace,
dis-grace,
grace-ful,
dis-grace-ful



FSA for Derivational Morphology

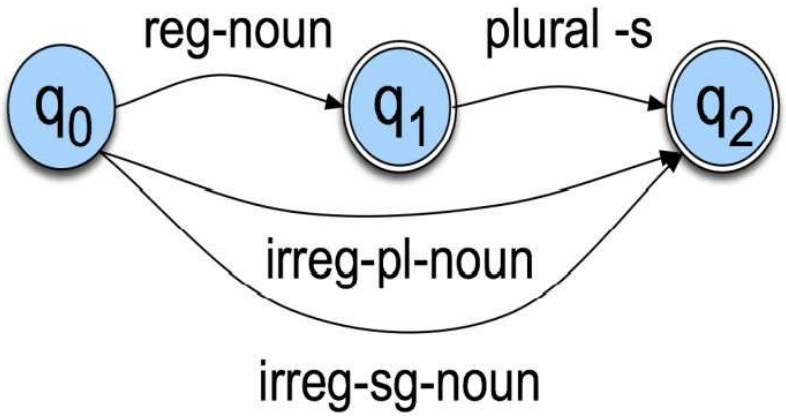


Irregular Words

Some irregular words require stem changes.

Past tense verbs:
teach-taught, go-went, write-wrote

Plural nouns:
mouse-mice, foot-feet, wife-wives



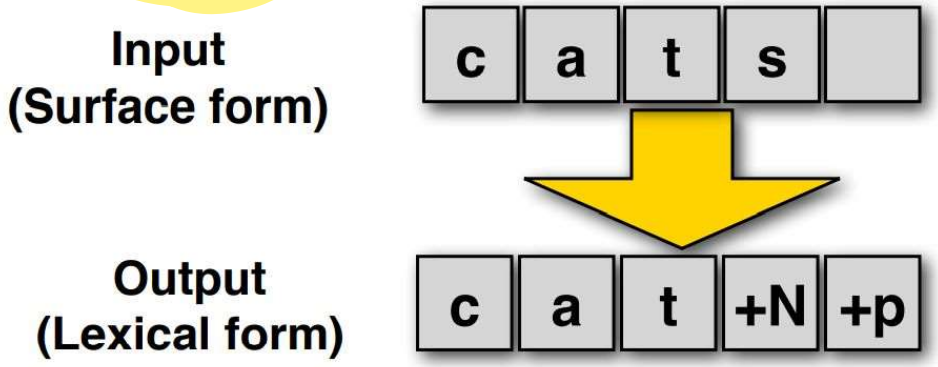
Recognition Vs Analysis

FSAs can recognize (accept) a string, but they don't tell us its internal structure.

fsa doesn't tell us about the structure of the sentence or the word , so we use fst

We need is a machine that maps (transduces) the input string into an output string that encodes its structure:

fst are used to find the morphemes



- FSA recognises a string but we do not anything about the internal structure of the string.

- FST ia actually a 2-tape FSA.FSTs have two input tapes and one of them can be read as an output tape.

- A finite state transducer defines relation between 2 regular languages Lin and Lout
  - L_{in} = {cat, cats, fox, foxes, ...}
  - L_{out} = {cat+N+sg, cat+N+pl, fox+N+sg, fox+N+PL ...}
  - T = { <cat, cat+N+sg>, <cats, cat+N+pl>, <fox, fox+N+sg>, <foxes, fox+N+pl> }

Finite State Transducer

A **finite-state transducer** $T = \langle Q, \Sigma, \Delta, q_0, F, \delta, \sigma \rangle$ consists of:

- A finite **set of states** $Q = \{q_0, q_1, .., q_n\}$
- A finite alphabet Σ of **input symbols** (e.g. $\Sigma = \{a, b, c, ...\}$)
- A finite alphabet Δ of **output symbols** (e.g. $\Delta = \{+N, +pl, ..\}$)
- A designated **start state** $q_0 \in Q$
- A set of **final states** $F \subseteq Q$
- A **transition function** $\delta: Q \times \Sigma \rightarrow 2^Q$

  $\delta(q,w) = Q'$ for $q \in Q$, $Q' \subseteq Q$, $w \in \Sigma$
- **An output function** $\sigma: Q \times \Sigma \rightarrow \Delta^*$

  $\sigma(q,w) = \omega$ for $q \in Q$, $w \in \Sigma$, $\omega \in \Delta^*$

  If the current state is q and the current input is w, write ω.

Finite State Transducers

An FST $T = L_{in} \times L_{out}$ defines a relation between two regular languages L_{in} and L_{out}:

L_{in} = {**cat, cats, fox, foxes, ...**}

L_{out} = {*cat+N+sg, cat+N+pl, fox+N+sg, fox+N+PL ...*}

T = { <**cat**, *cat+N+sg*>,
<**cats**, *cat+N+pl*>,
<**fox**, *fox+N+sg*>,
<**foxes**, *fox+N+pl*> }

Inversion is useful because it makes it easy to convert a FST-as-parser into an FSTas-generator.

Composition is useful because it allows us to take two transducers than run in series and replace them with one complex transducer.

Some FST Operations

Inversion T^{-1}:

The inversion (T^{-1}) of a transducer switches input and output labels.

This can be used to switch from parsing words to generating words.

Composition ($T \circ T'$): *(Cascade)*

Two transducers $T = L_1 \times L_2$ and $T' = L_2 \times L_3$ can be composed into a third transducer $T'' = L_1 \times L_3$.

Sometimes intermediate representations are useful

- Two tapes
  - Upper (**lexical**) tape: output alphabet Δ
    - cat +N +Pl
  - Lower (**surface**) tape: Input alphabet Σ
    - cats

Lexical		c	a	t	+N	+Pl	
Surface		c	a	t	s		

- English often requires spelling changes at morpheme boundaries
- Introduction of **orthographic rules**, as e.g.

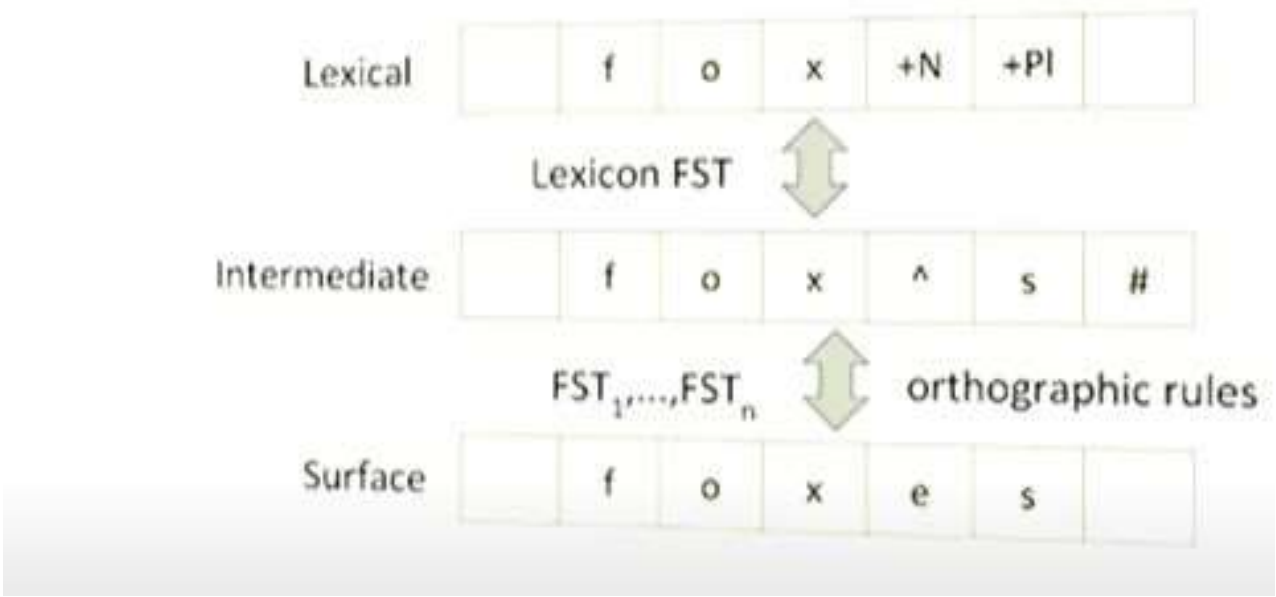
Name	Orthographic Rule	Example
Consonant doubling	Consonant doubled before -ing/-ed	beg / begging
E deletion	Silent e dropped before -ing and -ed	make / making
E insertion	E added after -s, -z, -x, -ch, -sh before -s	fox / foxes
Y replacement	-y changes to -ie before -s, -i before -ed	try / tries
K insertion	Verbs ending with vowel + -c ad -k	panic / panicked

KIT

- English plural -s:
  - cat ⇒ cats, dog ⇒ dogs
  - but: fox ⇒ foxes, buzz ⇒ buzzes
- We define an intermediate representation which captures **morpheme boundaries** (^) and **word boundaries** (#):

Lexicon:	**cat+N+PL**	**fox+N+PL**
Intermediate representation:	**cat^s#**	**fox^s#**
Surface string:	**cats**	**foxes**

- **Intermediate-to-Surface Spelling Rule:**
  If **plural 's'** follows a morpheme ending in **'x'**, **'z'** or **'s'**, ***insert 'e'***.


Lexical
f
o
x
+N
+Pl
Lexicon FST
Intermediate
f
o
x
^
s
#
FST1,...,FSTn
orthographic rules
Surface
f
o
x
e
s

Dealing with Ambiguity

book: *book +N +sg* or *book +V?*

Generating words is generally unambiguous, but analyzing words often requires disambiguation.