

# A very short introduction to Language Technologies and Natural Language Processing

Jose F Quesada & Jose Luis Pro

## Language

Set of conventional spoken or written symbols used for communication between entities.

So we can see a language as the linking between meaning (semantic side) and expression (syntactic side).

### Types of languages

- **Natural languages:** Used for the communication between human beings.
- **Formal languages:** Used by computers and in mathematical areas.

## Language

Set of conventional spoken or written symbols used for communication between entities.

So we can see a language as the linking between meaning (semantic side) and expression (syntactic side).

### Types of languages

- **Natural languages:** Used for the communication between human beings.
- **Formal languages:** Used by computers and in mathematical areas.

## Language

Set of conventional spoken or written symbols used for communication between entities.

So we can see a language as the linking between meaning (semantic side) and expression (syntactic side).

### Types of languages

- **Natural languages:** Used for the communication between human beings.
- **Formal languages:** Used by computers and in mathematical areas.

## Differences between natural and formal languages:

- 1 Computers don't understand natural languages, (normal) humans don't understand computer languages.
- 2 Formal languages shouldn't have ambiguities, but natural languages do have.

**Language Technologies** (LT's) are the set of technologies that aim to create software that has some kind of knowledge about natural languages.

**Natural language processing** (NLP) is the scientific field concerned with the interactions between computers and human by means of natural languages.

## Differences between natural and formal languages:

- 1 Computers don't understand natural languages, (normal) humans don't understand computer languages.
- 2 Formal languages shouldn't have ambiguities, but natural languages do have.

**Language Technologies** (LT's) are the set of technologies that aim to create software that has some kind of knowledge about natural languages.

**Natural language processing** (NLP) is the scientific field concerned with the interactions between computers and human by means of natural languages.

## Differences between natural and formal languages:

- 1 Computers don't understand natural languages, (normal) humans don't understand computer languages.
- 2 Formal languages shouldn't have ambiguities, but natural languages do have.

**Language Technologies** (LT's) are the set of technologies that aim to create software that has some kind of knowledge about natural languages.

**Natural language processing** (NLP) is the scientific field concerned with the interactions between computers and human by means of natural languages.

## Differences between natural and formal languages:

- 1 Computers don't understand natural languages, (normal) humans don't understand computer languages.
- 2 Formal languages shouldn't have ambiguities, but natural languages do have.

**Language Technologies** (LT's) are the set of technologies that aim to create software that has some kind of knowledge about natural languages.

**Natural language processing** (NLP) is the scientific field concerned with the interactions between computers and human by means of natural languages.



## Applications of Language Technologies:

- 1 Machine Translation.
- 2 Question - Answering.
- 3 Automatic Text Classification.
- 4 Automatic Text Summarization.
- 5 Social Analytics.
- 6 Sentiment Analysis.
- 7 **Dialogue Systems.**

## Applications of Language Technologies:

- 1 Machine Translation.
- 2 Question - Answering.
- 3 Automatic Text Classification.
- 4 Automatic Text Summarization.
- 5 Social Analytics.
- 6 Sentiment Analysis.
- 7 **Dialogue Systems.**

## Applications of Language Technologies:

- 1 Machine Translation.
- 2 Question - Answering.
- 3 Automatic Text Classification.
- 4 Automatic Text Summarization.
- 5 Social Analytics.
- 6 Sentiment Analysis.
- 7 **Dialogue Systems.**

## Applications of Language Technologies:

- 1 Machine Translation.
- 2 Question - Answering.
- 3 Automatic Text Classification.
- 4 Automatic Text Summarization.
- 5 Social Analytics.
- 6 Sentiment Analysis.
- 7 **Dialogue Systems.**

## Applications of Language Technologies:

- 1 Machine Translation.
- 2 Question - Answering.
- 3 Automatic Text Classification.
- 4 Automatic Text Summarization.
- 5 Social Analytics.
- 6 Sentiment Analysis.
- 7 **Dialogue Systems.**

## Applications of Language Technologies:

- 1 Machine Translation.
- 2 Question - Answering.
- 3 Automatic Text Classification.
- 4 Automatic Text Summarization.
- 5 Social Analytics.
- 6 Sentiment Analysis.
- 7 Dialogue Systems.

## Applications of Language Technologies:

- 1 Machine Translation.
- 2 Question - Answering.
- 3 Automatic Text Classification.
- 4 Automatic Text Summarization.
- 5 Social Analytics.
- 6 Sentiment Analysis.
- 7 **Dialogue Systems.**

Dialogue Systems (written or spoken) are also known as:

- Conversational interfaces.
- Chatbots.

### Applications of Dialogue Systems:

- 1 Information providers.
- 2 Transactional agents.
- 3 Educational and learning tutoring.



Dialogue Systems (written or spoken) are also known as:

- Conversational interfaces.
- Chatbots.

### Applications of Dialogue Systems:

- 1 Information providers.
- 2 Transactional agents.
- 3 Educational and learning tutoring.

Dialogue Systems (written or spoken) are also known as:

- Conversational interfaces.
- Chatbots.

### Applications of Dialogue Systems:

- 1 Information providers.
- 2 Transactional agents.
- 3 Educational and learning tutoring.

Dialogue Systems (written or spoken) are also known as:

- Conversational interfaces.
- Chatbots.

### Applications of Dialogue Systems:

- 1 Information providers.
- 2 Transactional agents.
- 3 Educational and learning tutoring.

## Dialogue systems main issue

The most difficult challenge in the design of conversational interfaces are related with the highly ambiguous nature of spoken languages.

### Example

Peter come yesterday.  
Yesterday Peter come.

Two syntatic expressions  $\iff$  One semantic form

## Dialogue systems main issue

The most difficult challenge in the design of conversational interfaces are related with the highly ambiguous nature of spoken languages.

## Example

Peter come yesterday.  
Yesterday Peter come.

Two syntatic expressions  $\iff$  One semantic form

## Dialogue systems main issue

The most difficult challenge in the design of conversational interfaces are related with the highly ambiguous nature of spoken languages.

## Example

Peter come yesterday.  
Yesterday Peter come.

Two syntatic expressions  $\iff$  One semantic form

## Example

Peter said John came yesterday.

- Was it yesterday when Peter said that?
- Was it yesterday when John came?

One syntactic expression  $\iff$  Two semantic forms

## Example

Peter said John came yesterday.

- Was it yesterday when Peter said that?
- Was it yesterday when John came?

One syntactic expression  $\iff$  Two semantic forms



## Example

Peter said John came yesterday.

- Was it yesterday when Peter said that?
- Was it yesterday when John came?

One syntactic expression  $\iff$  Two semantic forms

Humans can deal with these ambiguities applying what is called “psicolinguistic preferences” and, of course, logic and common-sense reasoning:

### Example

Peter said John will come yesterday.

From the computer point of view this sentence is such ambiguous like previous one but humans know that nobody “*will come yesterday*”.

Humans can deal with these ambiguities applying what is called “psicolinguistic preferences” and, of course, logic and common-sense reasoning:

### Example

Peter said John will come yesterday.

From the computer point of view this sentence is such ambiguous like previous one but humans know that nobody “*will come yesterday*”.

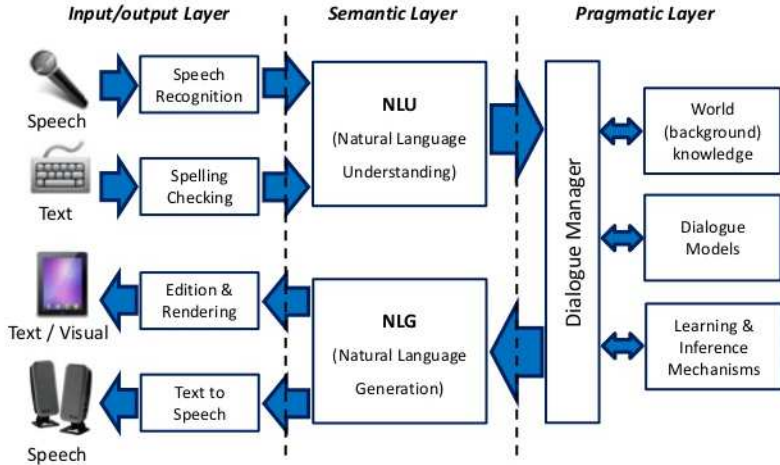
Humans can deal with these ambiguities applying what is called “psicolinguistic preferences” and, of course, logic and common-sense reasoning:

### Example

Peter said John will come yesterday.

From the computer point of view this sentence is such ambiguous like previous one but humans know that nobody “*will come yesterday*”.

# Dialogue System architecture



Picture from Institute for Infocomm Research (Singapore)

# Natural Language Understanding (NLU)

## NLU main goal

The goal of NLU stage is to transform an input string (let's say *user preference*) in an abstract representation of its meaning easier for computer programs to manipulate it, in order to execute some kind of reasoning.

There is a wide variety of possible meaning representations.

- Topic maps.
- Concepts maps.
- Mind maps.
- Ontologies.
- **Feature structures.**

# Natural Language Understanding (NLU)

## NLU main goal

The goal of NLU stage is to transform an input string (let's say *user preference*) in an abstract representation of its meaning easier for computer programs to manipulate it, in order to execute some kind of reasoning.

There is a wide variety of possible meaning representations.

- Topic maps.
- Concepts maps.
- Mind maps.
- Ontologies.
- **Feature structures.**

## Example

John came yesterday  $\rightarrow$   $\left[ \begin{array}{ll} \text{SUBJECT:} & \text{John} \\ \text{ACTION:} & \text{come} \\ \text{TENSE:} & \text{past} \\ \text{OFFSETDATE:} & -1 \text{ day} \end{array} \right]$

## Example

John will talk in two days  $\rightarrow$   $\left[ \begin{array}{ll} \text{SUBJECT:} & \text{John} \\ \text{ACTION:} & \text{talk} \\ \text{TENSE:} & \text{future} \\ \text{OFFSETDATE:} & +2 \text{ day} \end{array} \right]$



## Feature structures

- A feature structure is a set of features.
- With no particular order between them.
- Every feature may have (but it's not required) an associated value.
- The value associated to every feature can be **atomic** or **complex**.

comes  $\rightarrow$   $\left[ \begin{array}{ll} \text{ACTION:} & \text{come} \\ \text{TENSE:} & \text{present} \\ \text{AGREEMENT:} & \left[ \begin{array}{ll} \text{NUMBER:} & \text{singular} \\ \text{PERSON:} & 3 \end{array} \right] \end{array} \right]$

## Feature structures

- A feature structure is a set of features.
- With no particular order between them.
- Every feature may have (but it's not required) an associated value.
- The value associated to every feature can be **atomic** or **complex**.

comes  $\rightarrow$   $\left[ \begin{array}{ll} \text{ACTION:} & \text{come} \\ \text{TENSE:} & \text{present} \\ \text{AGREEMENT:} & \left[ \begin{array}{ll} \text{NUMBER:} & \text{singular} \\ \text{PERSON:} & 3 \end{array} \right] \end{array} \right]$

## NLU components

Trying to convert user preference to feature structures is not trivial.  
So we need to divide the process in some functional modules:

- Tokenization.
- Speller checker.
- Part Of Speech tagging (POS tagging).
- Parsing.
- Unifier.

# NLU components

Trying to convert user preference to feature structures is not trivial.  
So we need to divide the process in some functional modules:

- Tokenization.
- Speller checker.
- Part Of Speech tagging (POS tagging).
- Parsing.
- Unifier.

# Tokenization

## Goal

Convert a sequence of characters into a sequence of tokens.

We must take into account:

- Separators: ( \_ - \_ )
- Punctuation marks: ( , . ; : ! ? )
- Special symbols: ( \$ € % ° )
- Numbers and its own separators: ( 1234 , . )
- Alphanumeric codes: ( ES772024 . . . )

## Example

	tk1		tk2		tk3		tk4		tk5		tk6	
.	The	.	dog	.	is	.	in	.	the	.	park	.
0		1		2		3		4		5		6

# Tokenization

## Goal

Convert a sequence of characters into a sequence of tokens.

We must take into account:

- Separators: ( \_ - \_ )
- Punctuation marks: ( , . ; : ! ? )
- Special symbols: ( \$ € % ° )
- Numbers and its own separators: ( 1234 , . )
- Alphanumeric codes: ( ES772024 . . . )

## Example

	tk1		tk2		tk3		tk4		tk5		tk6	
.	The	.	dog	.	is	.	in	.	the	.	park	.
0		1		2		3		4		5		6

# Tokenization

## Goal

Convert a sequence of characters into a sequence of tokens.

We must take into account:

- Separators: ( \_ - \_ )
- Punctuation marks: ( , . ; : ! ? )
- Special symbols: ( \$ € % ° )
- Numbers and its own separators: ( 1234 , . )
- Alphanumeric codes: ( ES772024 . . . )

## Example

	tk1		tk2		tk3		tk4		tk5		tk6	
.	The	.	dog	.	is	.	in	.	the	.	park	.
0		1		2		3		4		5		6

## Speller checker (only in written dialogue systems)

London

- Insertion: Loondon
- Deletion: Lndon
- Substitution: Lpndon
- Switching: Lonodn
- Bad separators: Lon don



## Part Of Speech tagging (POS tagging)

### Goal

To mark up lexical items with some lexical category depending on its definition and the context.

In natural language we can have some common lexical categories:

- Determiners: a, the
- Nouns: London, dog
- Pronouns: you, me
- Prepositions: to, for
- Adjectives: blue, long

## Part Of Speech tagging (POS tagging)

### Goal

To mark up lexical items with some lexical category depending on its definition and the context.

In natural language we can have some common lexical categories:

- Determiners: a, the
- Nouns: London, dog
- Pronouns: you, me
- Prepositions: to, for
- Adjectives: blue, long

## POS tagging

So in the lexicon definition we can classify lexical items into categories:

- ("the", det)
- ("dog", noun)
- ("me", pronoun)
- ("to", preposition)

But in natural languages we can have several lexical categories corresponding to a single lexical item (especially in little inflectional ones, like in english):

- ("plans", noun) → plural of plan.
- ("plans", verb) → present of third person (singular) of verb to plan.

## POS tagging

So in the lexicon definition we can classify lexical items into categories:

- ("the", det)
- ("dog", noun)
- ("me", pronoun)
- ("to", preposition)

But in natural languages we can have several lexical categories corresponding to a single lexical item (especially in little inflectional ones, like in english):

- ("plans", noun) → plural of plan.
- ("plans", verb) → present of third person (singular) of verb to plan.

## POS tagging: Garden path problem

### Example

<u>The</u>	<u>government</u>	<u>plans</u>	<u>to</u>	<u>raise</u>	<u>taxes</u>	...
det	noun	<b>verb</b>	prep	verb	noun	

### Example

<u>The</u>	<u>government</u>	<u>plans</u>	<u>to</u>	<u>raise</u>	<u>taxes</u>	<u>were</u>	<u>defeated</u>
det	noun	<b>noun</b>	prep	verb	noun	verb	adj

## POS tagging: Garden path problem

### Example

<u>The</u>	<u>government</u>	<u>plans</u>	<u>to</u>	<u>raise</u>	<u>taxes</u>	...
det	noun	<b>verb</b>	prep	verb	noun	

### Example

<u>The</u>	<u>government</u>	<u>plans</u>	<u>to</u>	<u>raise</u>	<u>taxes</u>	<u>were</u>	<u>defeated</u>
det	noun	<b>noun</b>	prep	verb	noun	verb	adj

# Formal grammars

- Only study the purely syntactical aspects of language.
- **Alphabet:** Set of terminal symbols.  $\{a, b\}$
- **Sentences:** Strings of symbols.  $a, b, aa, ab, bb, ba, aaa, \dots$
- **Language:** Set of sentences.  $L = \{ab, aabb, aaabbb\}$
- **Formal grammar:** Set of formation rules used to define a language.  $\{S \rightarrow aA, A \rightarrow bA, S \rightarrow \varepsilon \dots\}$  Where  $S, A$  are non-terminal symbols.
- Depending on the syntax of production rules, grammars can be classified (Noam Chomsky, 1956).

# Chomsky grammar hierarchy

Type	Grammar accepted	Rules	Observations
Type 0	Unrestricted grammar	$X \rightarrow Y$	$X, Y \in (N \cup T)^*$
Type 1	Context-sensitive grammar	$\alpha X \beta \rightarrow \alpha a \beta$	$X \in N$ $\alpha, a, \beta \in (N \cup T)^*$
Type 2	Context-free grammar	$X \rightarrow a$	$X \in N$ $a \in (N \cup T)^*$
Type 3	Regular grammar	$X \rightarrow a$ $X \rightarrow aY$	$X, Y \in N$ $X \in T$

Where:

- $N$  is the set of non-terminal symbols.
- $T$  is the set of terminal symbols.
- $S^*$  is a string of elements in set  $S$ .



## Easy context-free grammar example: $a^n b^m$

Consider:

- **Alphabet:**  $\{a, b\}$
- **Language:**  $L = \{a^n b^m\} \ n, m \geq 1$  (i.e. all strings with at least one “a” followed by at least one “b”).

$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

- **Grammar:**  $A \rightarrow a A$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

And let's see that a grammar defined with such production rules can be used to **generate** or either **recognize** the target language.

# Generating $a^n b^m$ sentences

$S$

$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

# Generating $a^n b^m$ sentences

S

$$S \rightarrow a A b B$$

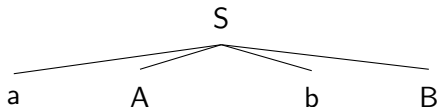
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

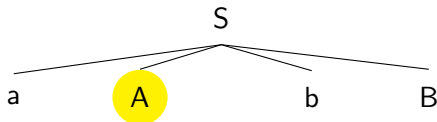
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

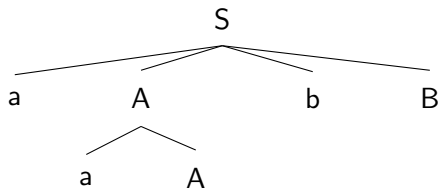
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

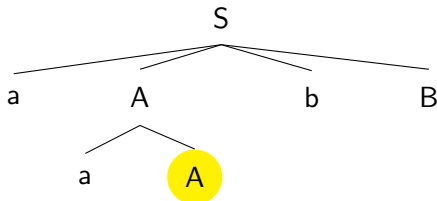
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

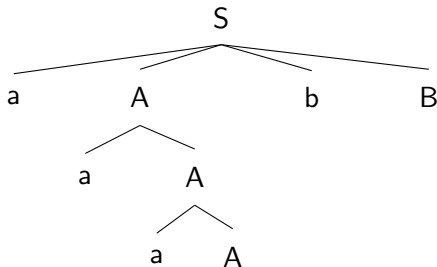
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

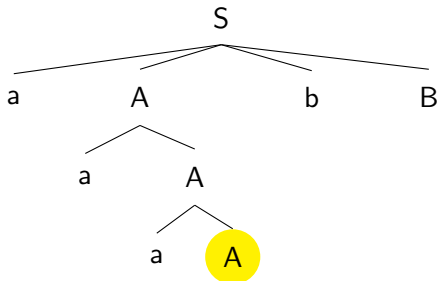
$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$



## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

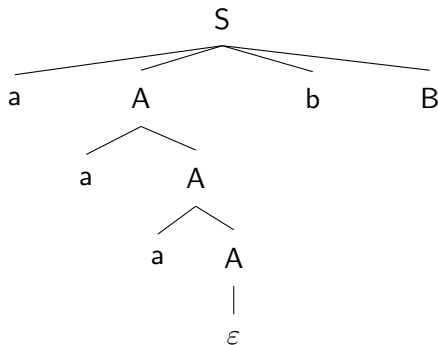
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

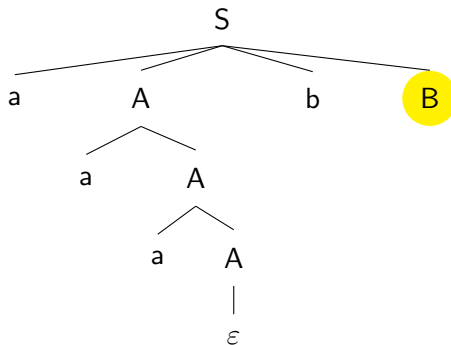
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

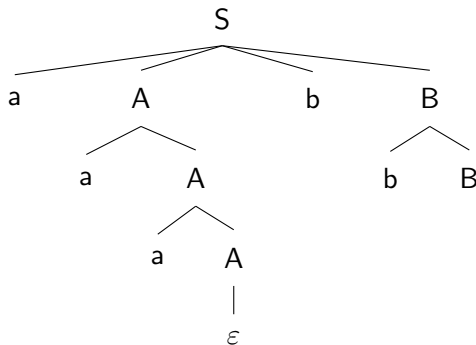
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

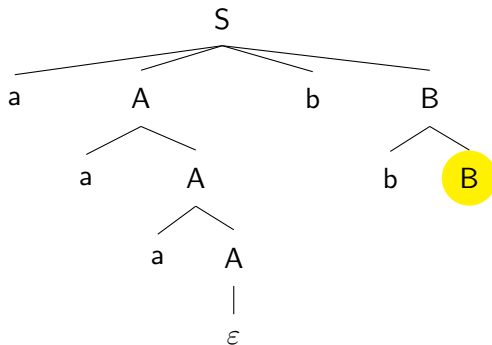
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Generating $a^n b^m$ sentences



$$S \rightarrow a A b B$$

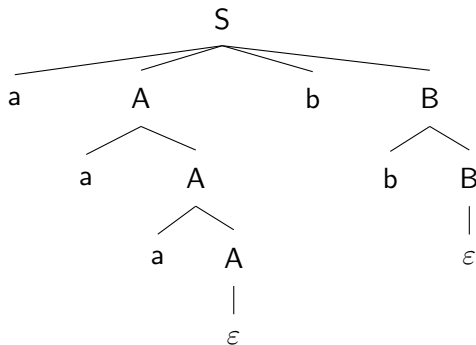
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

So we have generated *aaabb* sentence!



$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Recognizing *aaabb* sentence

a

b

a

b

a

$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

## Recognizing *aaabb* sentence

a

b

a

b

a

$\epsilon$

$\epsilon$

$$S \rightarrow a A b B$$

$$A \rightarrow \epsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \epsilon$$

$$B \rightarrow b B$$



## Recognizing *aaabb* sentence

a

b

a

b

a

$\epsilon$

$\epsilon$

$$S \rightarrow a A b B$$

$$A \rightarrow \epsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \epsilon$$

$$B \rightarrow b B$$

# Recognizing *aaabb* sentence

a

b

a

b

B

a

A

$\varepsilon$

|

$\varepsilon$

$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

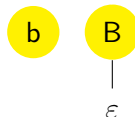
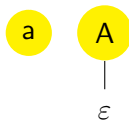
$$B \rightarrow b B$$

# Recognizing *aaabb* sentence

a

b

a



$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

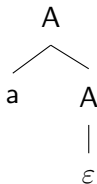
$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

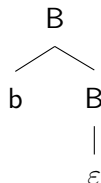
## Recognizing *aaabb* sentence

a

a



b



$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

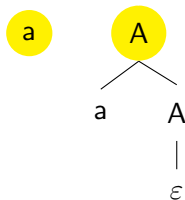
$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

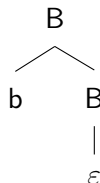
$$B \rightarrow b B$$

## Recognizing *aaabb* sentence

a



b



$$S \rightarrow a A b B$$

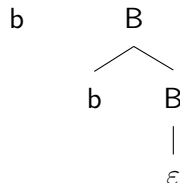
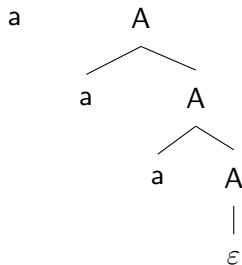
$$A \rightarrow \epsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \epsilon$$

$$B \rightarrow b B$$

## Recognizing *aaabb* sentence



$$S \rightarrow a A b B$$

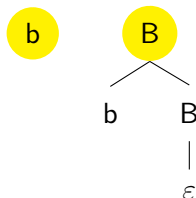
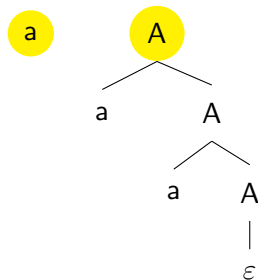
$$A \rightarrow \epsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \epsilon$$

$$B \rightarrow b B$$

## Recognizing *aaabb* sentence



$$S \rightarrow a A b B$$

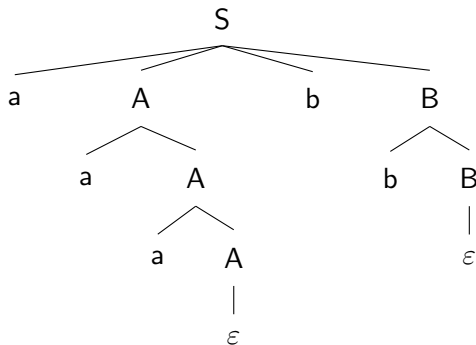
$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$

We have recognized *aaabb* sentence!



$$S \rightarrow a A b B$$

$$A \rightarrow \varepsilon$$

$$A \rightarrow a A$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow b B$$



## Parsing

- Is the process of analysing a sentence, according to the rules of a certain grammar, resulting in the creation of a “parse tree”.
- That parse tree includes syntactic relations between components in the sentence but may also contains semantic information (e.g. a feature structure).

## Parsing

- Is the process of analysing a sentence, according to the rules of a certain grammar, resulting in the creation of a “parse tree”.
- That parse tree includes syntactic relations between components in the sentence but may also contains semantic information (e.g. a feature structure).

## Parsing

- Is the process of analysing a sentence, according to the rules of a certain grammar, resulting in the creation of a “parse tree”.
- That parse tree includes syntactic relations between components in the sentence but may also contains semantic information (e.g. a feature structure).



CC by Kat at the English language Wikipedia SA. 3.0

# Parsing applied to natural languages (english)

Some typical lexical categories:

- det: Determiners.
- noun: Nouns.
- verb: Verbs.
- prep: Prepositions.

Some typical non-terminal symbols:

- S: Sentence.
- NP: Noun Phrase.
- VP: Verb Phrase.
- PP: Prepositional Phrase.

Some typical production rules

- $S \rightarrow NP VP$
- $NP \rightarrow \text{det noun}$
- $VP \rightarrow \text{verb}$
- $VP \rightarrow \text{verb NP}$
- $PP \rightarrow \text{prep NP}$
- $NP \rightarrow NP PP$
- $VP \rightarrow VP PP$

# Parsing applied to natural languages (english)

Some typical lexical categories:

- det: Determiners.
- noun: Nouns.
- verb: Verbs.
- prep: Prepositions.

Some typical non-terminal symbols:

- S: Sentence.
- NP: Noun Phrase.
- VP: Verb Phrase.
- PP: Prepositional Phrase.

Some typical production rules

- $S \rightarrow NP VP$
- $NP \rightarrow \text{det noun}$
- $VP \rightarrow \text{verb}$
- $VP \rightarrow \text{verb NP}$
- $PP \rightarrow \text{prep NP}$
- $NP \rightarrow NP PP$
- $VP \rightarrow VP PP$

# Parsing applied to natural languages (english)

Some typical lexical categories:

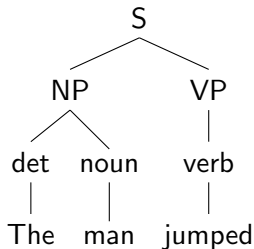
- det: Determiners.
- noun: Nouns.
- verb: Verbs.
- prep: Prepositions.

Some typical non-terminal symbols:

- S: Sentence.
- NP: Noun Phrase.
- VP: Verb Phrase.
- PP: Prepositional Phrase.

Some typical production rules

- $S \rightarrow NP VP$
- $NP \rightarrow \text{det noun}$
- $VP \rightarrow \text{verb}$
- $VP \rightarrow \text{verb NP}$
- $PP \rightarrow \text{prep NP}$
- $NP \rightarrow NP PP$
- $VP \rightarrow VP PP$



$S \rightarrow NP VP$

$NP \rightarrow \text{det noun}$

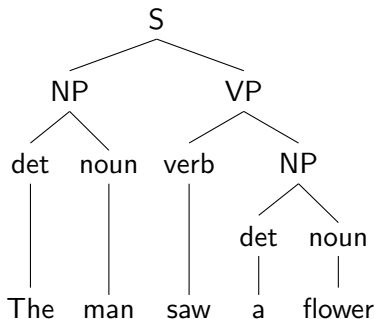
$VP \rightarrow \text{verb}$

$VP \rightarrow \text{verb NP}$

$PP \rightarrow \text{prep NP}$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$



$S \rightarrow NP VP$

$NP \rightarrow \text{det noun}$

$VP \rightarrow \text{verb}$

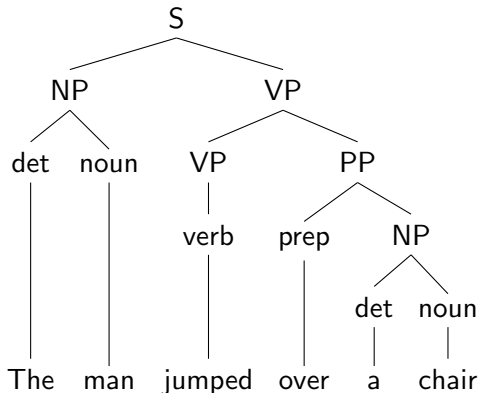
$VP \rightarrow \text{verb NP}$

$PP \rightarrow \text{prep NP}$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$





$S \rightarrow NP VP$

$NP \rightarrow \text{det noun}$

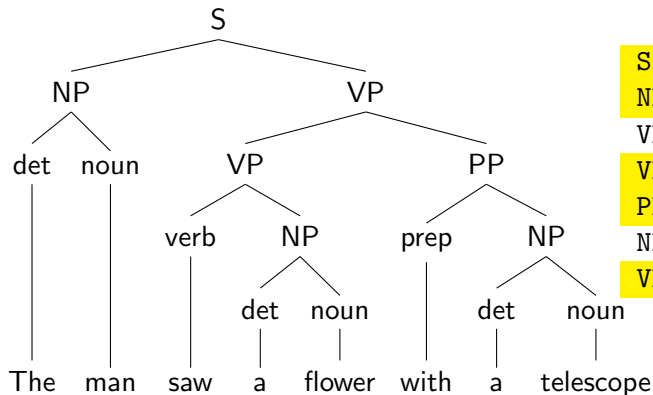
$VP \rightarrow \text{verb}$

$VP \rightarrow \text{verb NP}$

$PP \rightarrow \text{prep NP}$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$



$S \rightarrow NP VP$

$NP \rightarrow \text{det noun}$

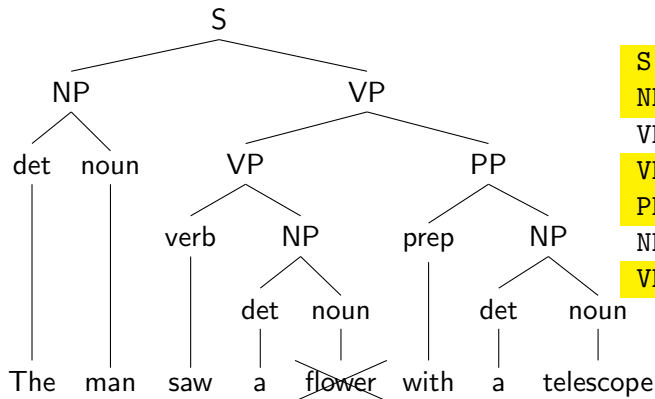
$VP \rightarrow \text{verb}$

$VP \rightarrow \text{verb NP}$

$PP \rightarrow \text{prep NP}$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$



$S \rightarrow NP VP$

$NP \rightarrow \text{det noun}$

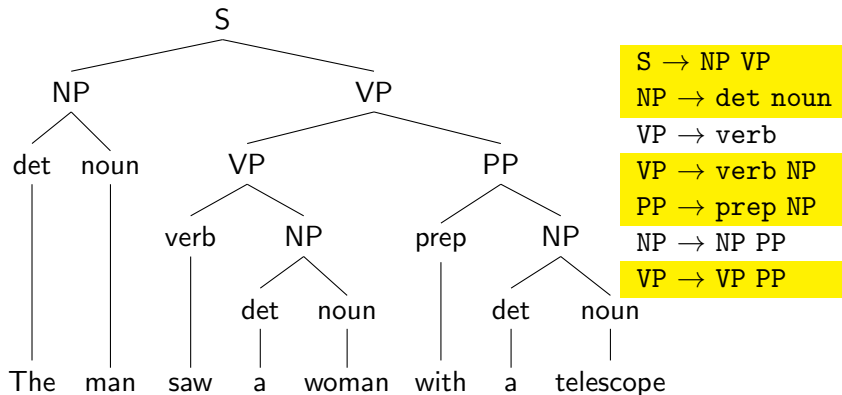
$VP \rightarrow \text{verb}$

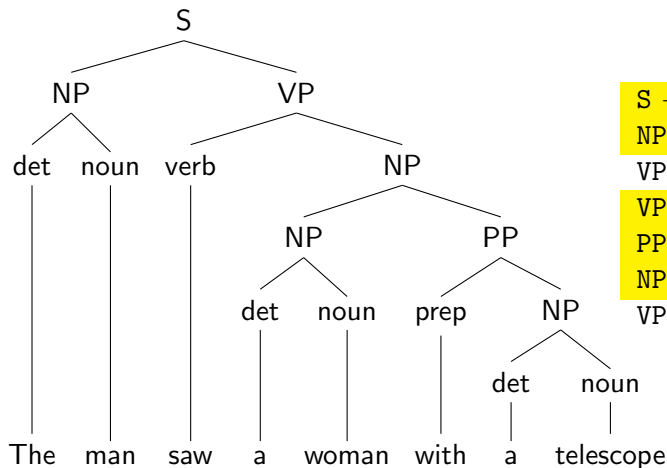
$VP \rightarrow \text{verb NP}$

$PP \rightarrow \text{prep NP}$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$





$S \rightarrow NP VP$

$NP \rightarrow \text{det noun}$

$VP \rightarrow \text{verb}$

$VP \rightarrow \text{verb NP}$

$PP \rightarrow \text{prep NP}$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

## Conclusions

- With so little production rules we have ambiguities.
- In practice, creating a general grammar results in too much ambiguities and poor performance.
- **Solution:** Create simple grammars for simple parameter extraction.

## Conclusions

- With so little production rules we have ambiguities.
- In practice, creating a general grammar results in too much ambiguities and poor performance.
- **Solution:** Create simple grammars for simple parameter extraction.

## Conclusions

- With so little production rules we have ambiguities.
- In practice, creating a general grammar results in too much ambiguities and poor performance.
- **Solution:** Create simple grammars for simple parameter extraction.



## Conclusions

- With so little production rules we have ambiguities.
- In practice, creating a general grammar results in too much ambiguities and poor performance.
- **Solution:** Create simple grammars for simple parameter extraction.



CC by Kat at the English language Wikipedia SA. 3.0

# Unification

- The main goal of unification is to find out that two different terms are in fact identical.
- Using unification **with** feature structures (Lexical Object Theory - Jose F Quesada):
  - Adequate for linguistic phenomena.
  - Computationally efficient.
  - Formal soundness.

$$\begin{bmatrix} \text{SUBJECT:} & [\text{NUMBER: plural}] \\ \text{SUBJECT:} & [\text{PERSON: 3}] \\ \text{TENSE:} & \text{present} \end{bmatrix} \quad \&gt; \quad \begin{bmatrix} \text{SUBJECT:} & [\text{NUMBER: plural} \\ & \text{PERSON: 3}] \\ \text{TENSE:} & \text{present} \end{bmatrix}$$

# Unification

- The main goal of unification is to find out that two different terms are in fact identical.
- Using unification **with** feature structures (Lexical Object Theory - Jose F Quesada):
  - Adequate for linguistic phenomena.
  - Computationally efficient.
  - Formal soundness.

$$\begin{bmatrix} \text{SUBJECT:} & [\text{NUMBER: plural}] \\ \text{SUBJECT:} & [\text{PERSON: 3}] \\ \text{TENSE:} & \text{present} \end{bmatrix} \quad \&gt; \quad \begin{bmatrix} \text{SUBJECT:} & [\text{NUMBER: plural} \\ & \text{PERSON: 3}] \\ \text{TENSE:} & \text{present} \end{bmatrix}$$

# Unification

- The main goal of unification is to find out that two different terms are in fact identical.
- Using unification **with** feature structures (Lexical Object Theory - Jose F Quesada):
  - Adequate for linguistic phenomena.
  - Computationally efficient.
  - Formal soundness.

[SUBJECT: [NUMBER: plural]]

[SUBJECT: [PERSON: 3]  
TENSE: present]

&>

[SUBJECT: [NUMBER: plural]  
PERSON: 3]  
TENSE: present]

# Unification

- The main goal of unification is to find out that two different terms are in fact identical.
- Using unification **with** feature structures (Lexical Object Theory - Jose F Quesada):
  - Adequate for linguistic phenomena.
  - Computationally efficient.
  - Formal soundness.

$$\begin{bmatrix} \text{SUBJECT:} & [\text{NUMBER: plural}] \\ \text{SUBJECT:} & [\text{PERSON: 3}] \\ \text{TENSE:} & \text{present} \end{bmatrix} \quad \&gt; \quad \begin{bmatrix} \text{SUBJECT:} & [\text{NUMBER: plural} \\ & \text{PERSON: 3}] \\ \text{TENSE:} & \text{present} \end{bmatrix}$$

[SUBJECT: [NUMBER: plural]]

[ TENSE: past  
SUBJECT: [NUMBER: singular]]

&>

FAIL!!

[SUBJECT: [PERSON: 3]  
TENSE: present]

[ TENSE: past  
SUBJECT: [NUMBER: singular]]

&>

FAIL!!

[SUBJECT: [NUMBER: plural]]

[ TENSE: past  
SUBJECT: [NUMBER: singular] ] &> FAIL!!

[SUBJECT: [PERSON: 3]  
TENSE: present ]

[ TENSE: past  
SUBJECT: [NUMBER: singular] ] &> FAIL!!

[SUBJECT: [NUMBER: plural]]

[ TENSE: past  
SUBJECT: [NUMBER: singular]]

&>

FAIL!!

[SUBJECT: [PERSON: 3]  
TENSE: present]

[ TENSE: past  
SUBJECT: [NUMBER: singular]]

&>

FAIL!!



[SUBJECT: [NUMBER: plural]]

[ TENSE: past  
SUBJECT: [NUMBER: singular] ] &> FAIL!!

[SUBJECT: [PERSON: 3]  
TENSE: present ]

[ TENSE: past  
SUBJECT: [NUMBER: singular] ] &> FAIL!!



CC by Kat at the English language Wikipedia SA. 3.0

# Dialogue Manager

## Definition of Dialogue Manager (DM)

It's the part of a dialogue system responsible for the state and flow of the conversation.

- The input of **DM** is the human preference converted to a semantic representation by **NLU** component.
- **DM** maintains some state variables (dialogue history, latest unanswered question, etc.) depending on the system (ie. the **context**).
- The output of **DM** will be another semantic representation of what system want to communicate to human. This will be converted to natural language by the **NLG** component.

# Dialogue Manager

## Definition of Dialogue Manager (**DM**)

It's the part of a dialogue system responsible for the state and flow of the conversation.

- The input of **DM** is the human preference converted to a semantic representation by **NLU** component.
- **DM** maintains some state variables (dialogue history, latest unanswered question, etc.) depending on the system (ie. the **context**).
- The output of **DM** will be another semantic representation of what system want to communicate to human. This will be converted to natural language by the **NLG** component.

# Pragmatics

- Pragmatics is a subfield in linguistics that studies the way in which **context** contributes to meaning.
- So pragmatics studies how the transmission of meaning depends not only on grammar or lexicon, but also on the context.
- These techniques explain how human beings are able to overcome apparent ambiguities or misunderstandings.

## Example

The man saw a woman with a telescope.

S: How old are you?

U: 24 → no need of confirmation

# Pragmatics

- Pragmatics is a subfield in linguistics that studies the way in which **context** contributes to meaning.
- So pragmatics studies how the transmission of meaning depends not only on grammar or lexicon, but also on the context.
- These techniques explain how human beings are able to overcome apparent ambiguities or misunderstandings.

## Example

The man saw a woman with a telescope.

S: How old are you?

U: 24 → no need of confirmation

# Pragmatics

- Pragmatics is a subfield in linguistics that studies the way in which **context** contributes to meaning.
- So pragmatics studies how the transmission of meaning depends not only on grammar or lexicon, but also on the context.
- These techniques explain how human beings are able to overcome apparent ambiguities or misunderstandings.

## Example

The man saw a woman with a telescope.

**S:** How old are you?

**U:** 24 → no need of confirmation

# Pragmatics

- Pragmatics is a subfield in linguistics that studies the way in which **context** contributes to meaning.
- So pragmatics studies how the transmission of meaning depends not only on grammar or lexicon, but also on the context.
- These techniques explain how human beings are able to overcome apparent ambiguities or misunderstandings.

## Example

The man saw a woman with a telescope.

**S:** How old are you?

**U:** 24 → no need of confirmation

# Pragmatics

- Pragmatics is a subfield in linguistics that studies the way in which **context** contributes to meaning.
- So pragmatics studies how the transmission of meaning depends not only on grammar or lexicon, but also on the context.
- These techniques explain how human beings are able to overcome apparent ambiguities or misunderstandings.

## Example

The man saw a woman with a telescope.

**S:** ~~How old are you?~~

**U:** 24



# Pragmatics

- Pragmatics is a subfield in linguistics that studies the way in which **context** contributes to meaning.
- So pragmatics studies how the transmission of meaning depends not only on grammar or lexicon, but also on the context.
- These techniques explain how human beings are able to overcome apparent ambiguities or misunderstandings.

## Example

The man saw a woman with a telescope.

**S:** How many pizzas do you want?

**U:** 24 → we seriously need to confirm this!

# Pragmatics

- Pragmatics is a subfield in linguistics that studies the way in which **context** contributes to meaning.
- So pragmatics studies how the transmission of meaning depends not only on grammar or lexicon, but also on the context.
- These techniques explain how human beings are able to overcome apparent ambiguities or misunderstandings.

## Example

The man saw a woman with a telescope.

**S:** How many pizzas do you want?

**U:** 24 → we seriously need to confirm this!

# Natural Language Generation (NLG)

- It's the last component of the Dialogue System.
- Receives as input some semantic representation (from **DM**) of what system must say.
- The output will be a string in natural language trying to communicate that semantic representation.
- Traditionally, the process is decomposed into two stages:
  - **Conceptualizer**: What to say.
  - **Formulator**: How to say.
- But frontiers between these modules are subtle: sequential, integrated, interactive, blackboard, revision-based,...

# Natural Language Generation (NLG)

- It's the last component of the Dialogue System.
- Receives as input some semantic representation (from **DM**) of what system must say.
- The output will be a string in natural language trying to communicate that semantic representation.
- Traditionally, the process is decomposed into two stages:
  - **Conceptualizer**: What to say.
  - **Formulator**: How to say.
- But frontiers between these modules are subtle: sequential, integrated, interactive, blackboard, revision-based,...

# Natural Language Generation (NLG)

- It's the last component of the Dialogue System.
- Receives as input some semantic representation (from **DM**) of what system must say.
- The output will be a string in natural language trying to communicate that semantic representation.
- Traditionally, the process is decomposed into two stages:
  - **Conceptualizer**: What to say.
  - **Formulator**: How to say.
- But frontiers between these modules are subtle: sequential, integrated, interactive, blackboard, revision-based,...

# Natural Language Generation (NLG)

- It's the last component of the Dialogue System.
- Receives as input some semantic representation (from **DM**) of what system must say.
- The output will be a string in natural language trying to communicate that semantic representation.
- Traditionally, the process is decomposed into two stages:
  - **Conceptualizer**: What to say.
  - **Formulator**: How to say.
- But frontiers between these modules are subtle: sequential, integrated, interactive, blackboard, revision-based,...

# Natural Language Generation (NLG)

- It's the last component of the Dialogue System.
- Receives as input some semantic representation (from **DM**) of what system must say.
- The output will be a string in natural language trying to communicate that semantic representation.
- Traditionally, the process is decomposed into two stages:
  - **Conceptualizer**: What to say.
  - **Formulator**: How to say.
- But frontiers between these modules are subtle: sequential, integrated, interactive, blackboard, revision-based,...

## Template-based systems

- But, in practice, a simple and efficient model for language generation is templated-based systems.
- These systems map their input (semantic representation) directly to linguistic structure. This structure may contain “gaps” which are filled in during output.

### Example

DEPARTURE:	[	FLIGHT:	Delta 2047	]
		GATE:	B37	
		TIME:	15:00	

{Flight} flight will depart from gate {Gate} at {Time}.



## Template-based systems

- But, in practice, a simple and efficient model for language generation is templated-based systems.
- These systems map their input (semantic representation) directly to linguistic structure. This structure may contain “gaps” which are filled in during output.

### Example

DEPARTURE:	[FLIGHT: Delta 2047]
	[GATE: B37]
	[TIME: 15:00]

{Flight} flight will depart from gate {Gate} at {Time}.

## Template-based systems

- But, in practice, a simple and efficient model for language generation is templated-based systems.
- These systems map their input (semantic representation) directly to linguistic structure. This structure may contain “gaps” which are filled in during output.

### Example

DEPARTURE:	[FLIGHT: Delta 2047]
	[GATE: B37]
	[TIME: 15:00]

{Flight} flight will depart from gate {Gate} at {Time}.



CC by Kat at the English language Wikipedia SA. 3.0