

Manuel Giuliani

Week 6

Human-Robot Interaction

Natural Language Processing

**UWE
Bristol**

University
of the
West of
England

Lectures overview

W.	Lecture
1	Introduction to HRI
2	Human factors and context
3	Design for HRI systems
4	User studies
5	Social signal processing
6	Natural language processing

W.	Lecture
7	Speech synthesis
8	Human-aware motion planning
9	Symbolic reasoning for HRI
10	Architecture for HRI
11	Statistics for HRI user studies
12	Exam revision

Learning outcomes

To know how speech recognition works

To know components for natural language processing and how they work

To know available NLP programs / libraries for your own HRI projects

Why do we need Natural Language Processing?

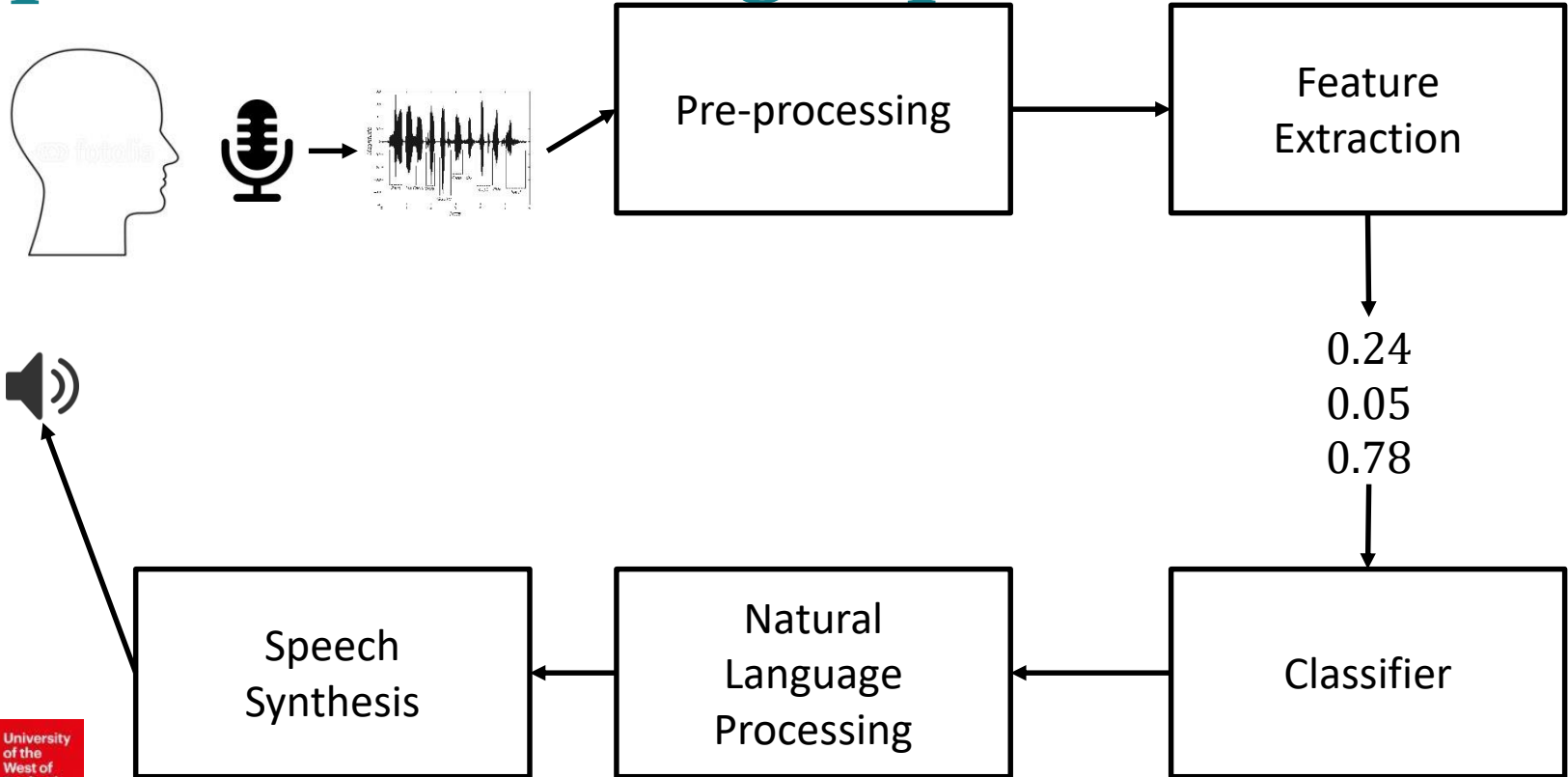


Why do we need NLP?

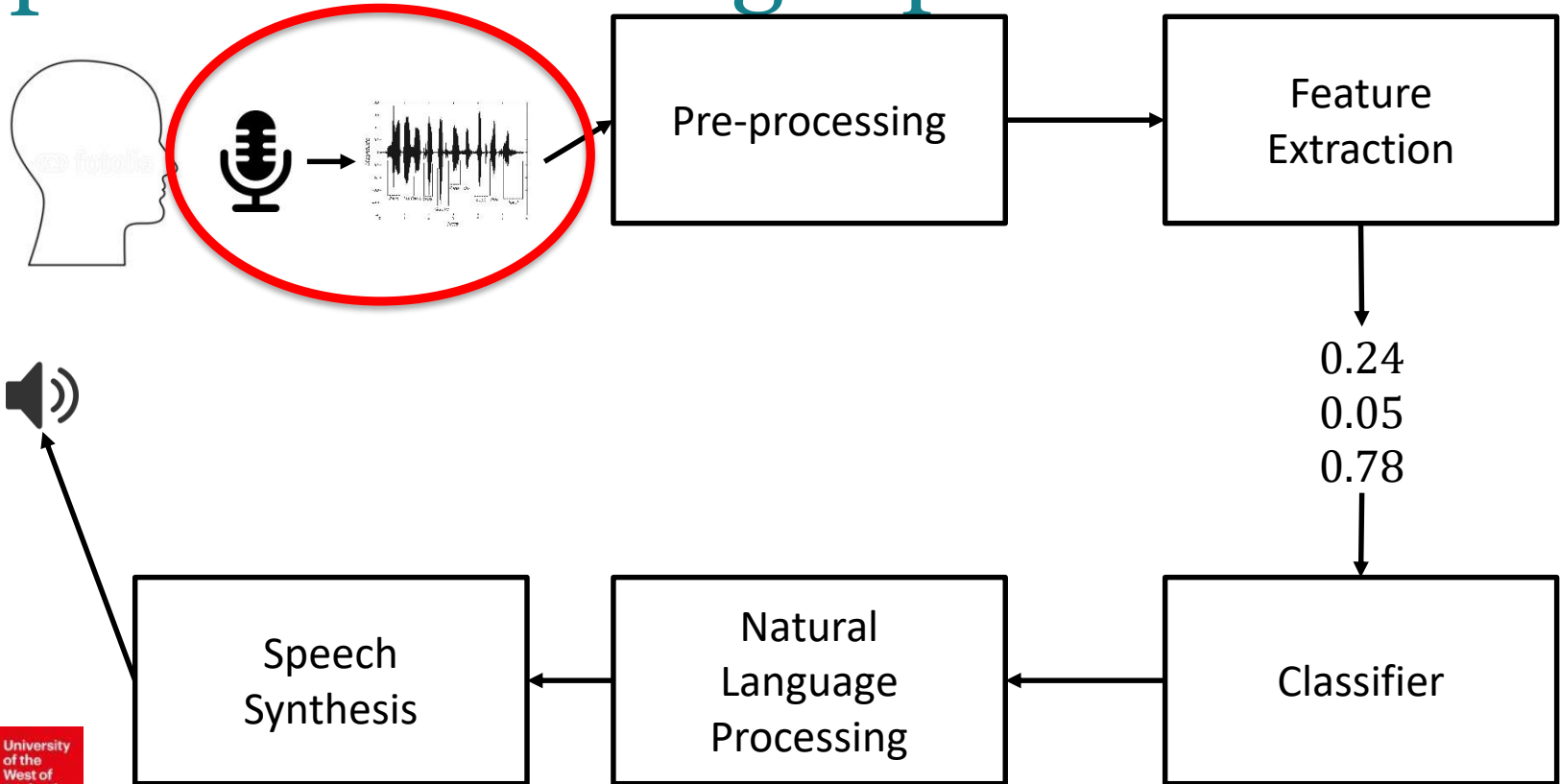
Language is generative. Theoretically one can generate infinitely many different sentences with every human language. Therefore, on a computer we need to work programs that can analyse human language and not just respond to a set of sentences.

Natural language processing programs are designed to break down language and to analyse their interrelations so that it is not necessary to collect an infinite number of language examples to understand what a person says.

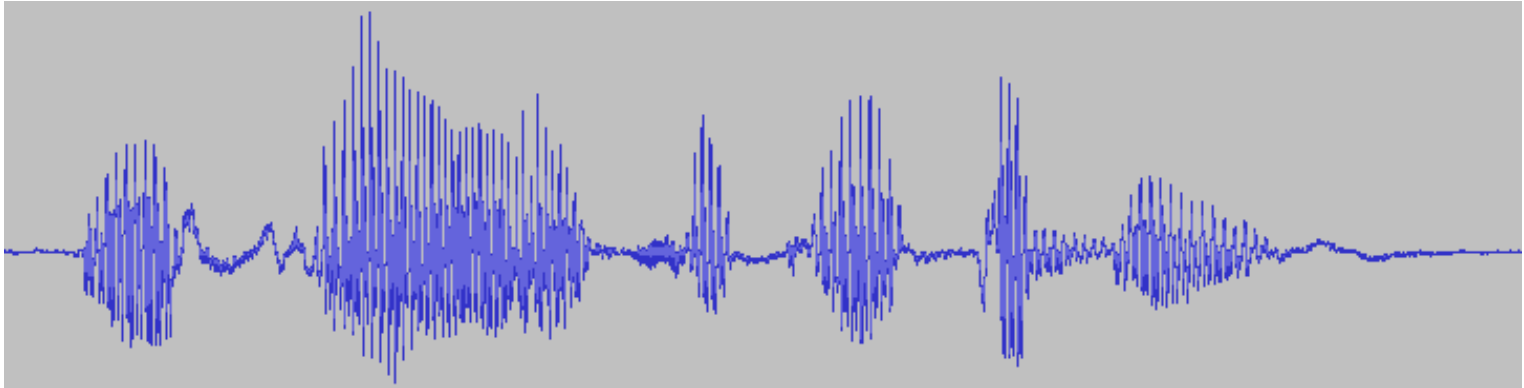
Speech Processing Pipeline



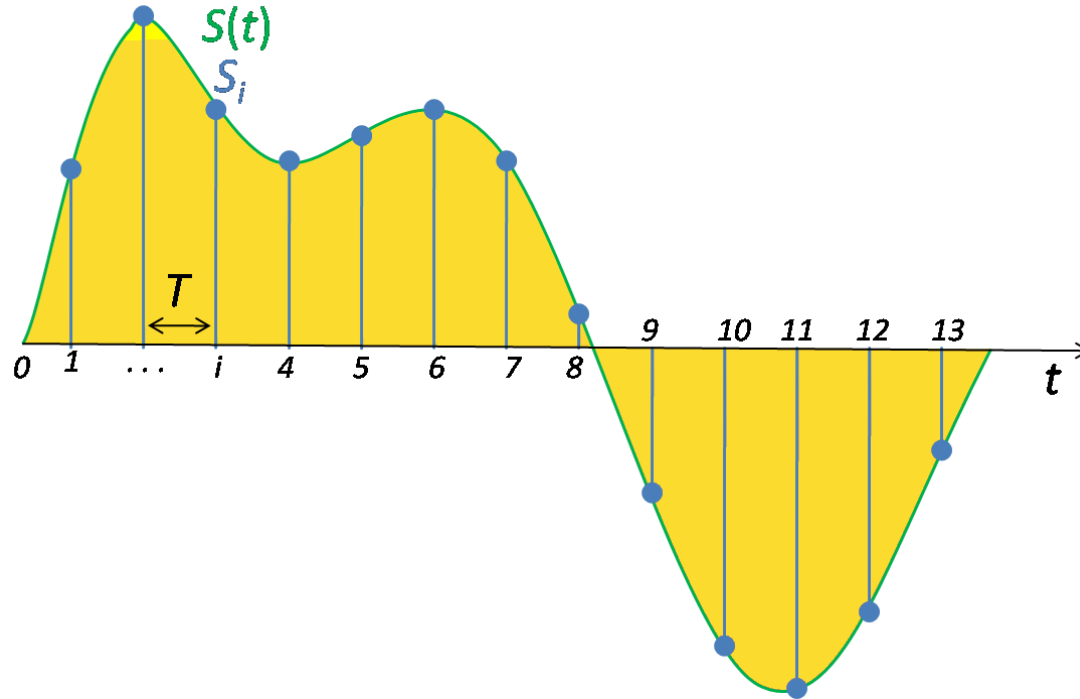
Speech Processing Pipeline



Sampling



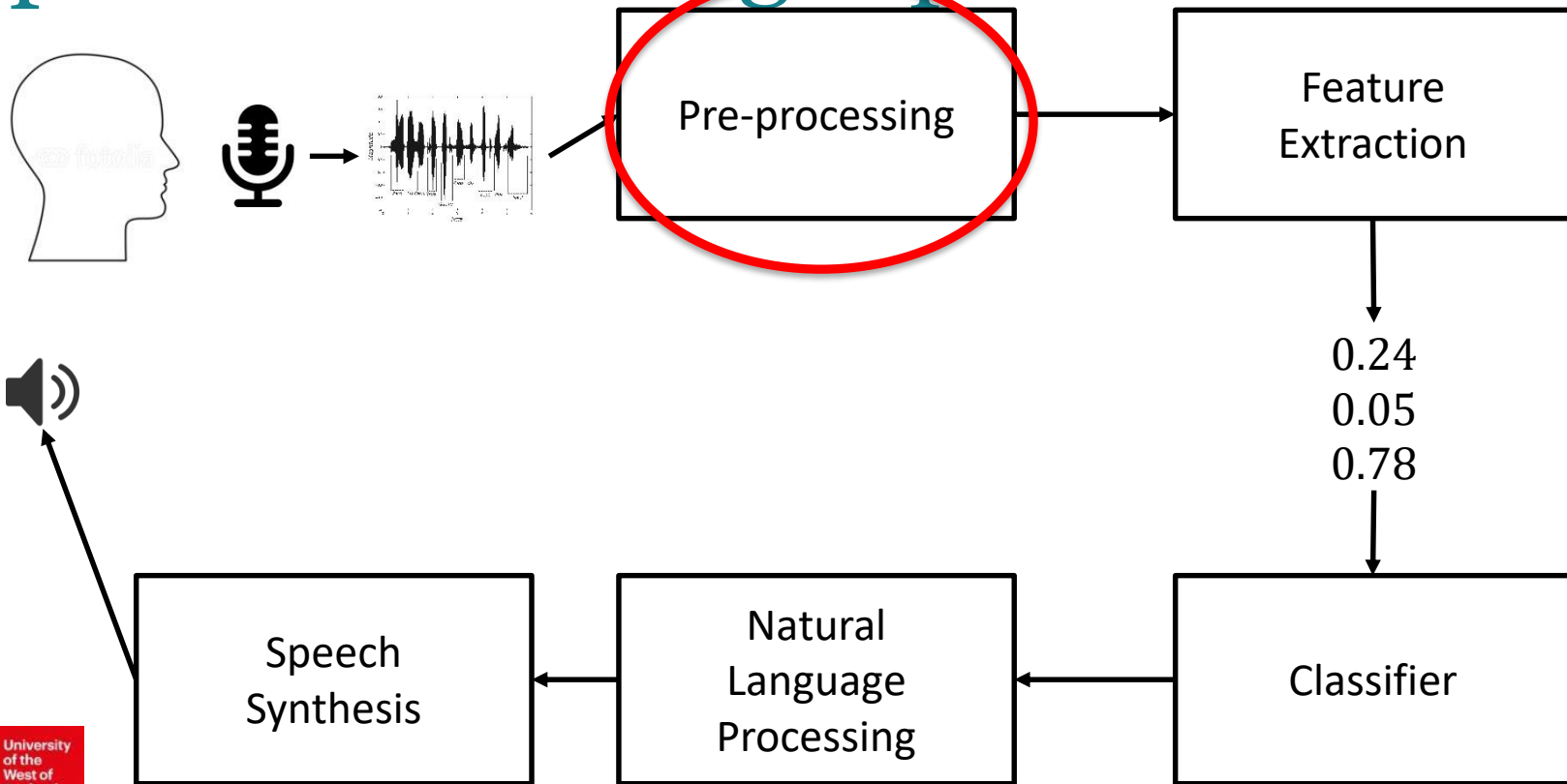
Sampling



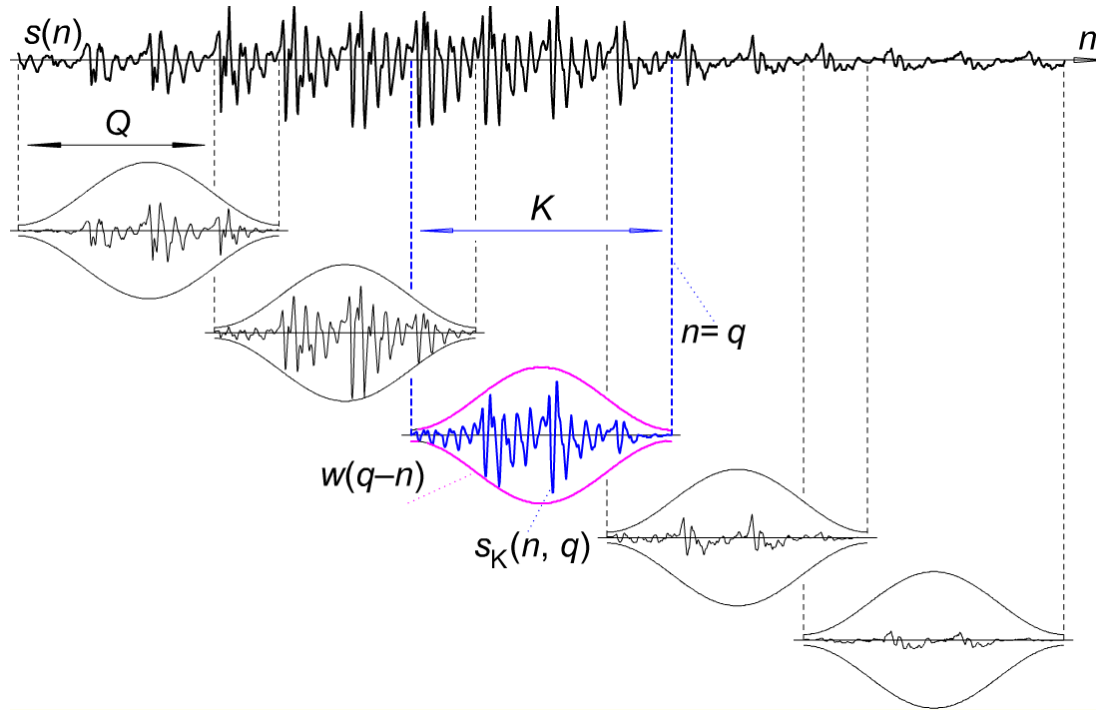
Sampling

- Digitizing the recorded speech signal
- Transition from continuous speech signal to time and value discrete signal
- For human speech a sampling frequency of 16kHz is sufficient (→ [Nyquist-Shannon Theorem](#))
- Each sample is quantified, depending on the computing power with 8 bits (= 256 values) or 16 bits (= 65536 values)

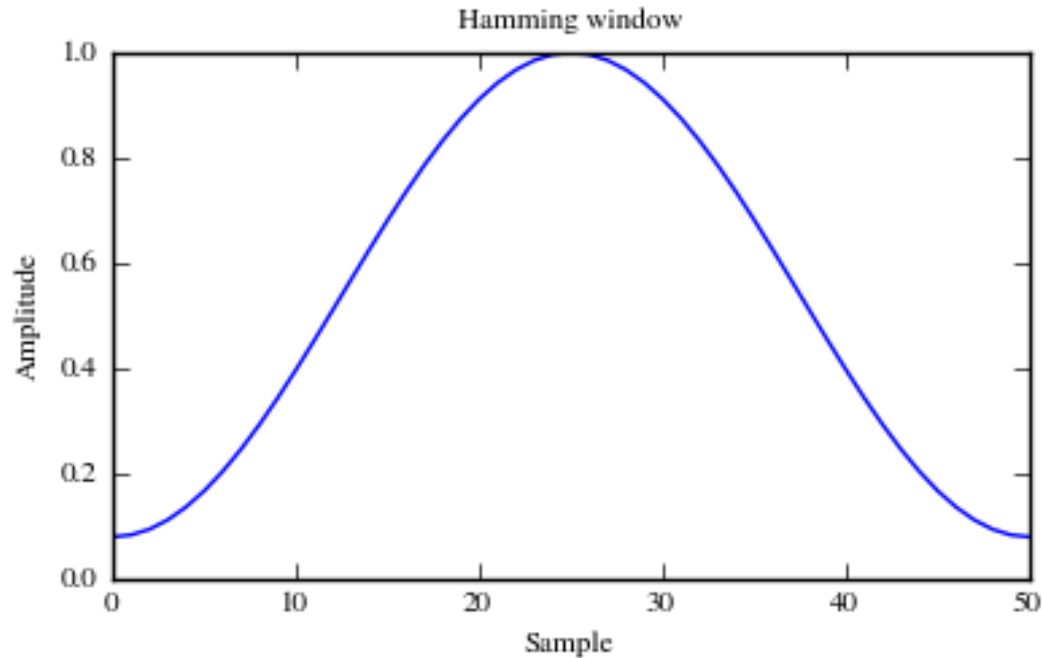
Speech Processing Pipeline



Pre-processing: Windowing



Hamming Window



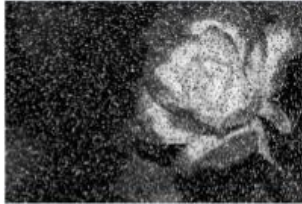
Pre-processing: Windowing

- To obtain individual frames (= sections) of the signal, the sections are multiplied by a windowing function.
- This way you get weighted sections that you can analyse further.
- Speech recognition typically uses windows between 5ms and 25ms.
- The shape of the windowing function is often chosen so that discontinuities on the edge of the window are muted.
- A frequently chosen windowing function is the Hamming Window.

Pre-processing: Denoising



Original



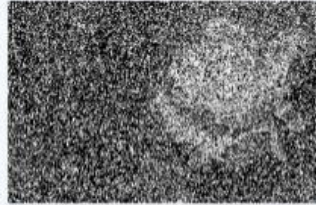
Salt & pepper
%20



De-noising by
Median filter



Original



Salt & pepper
%60

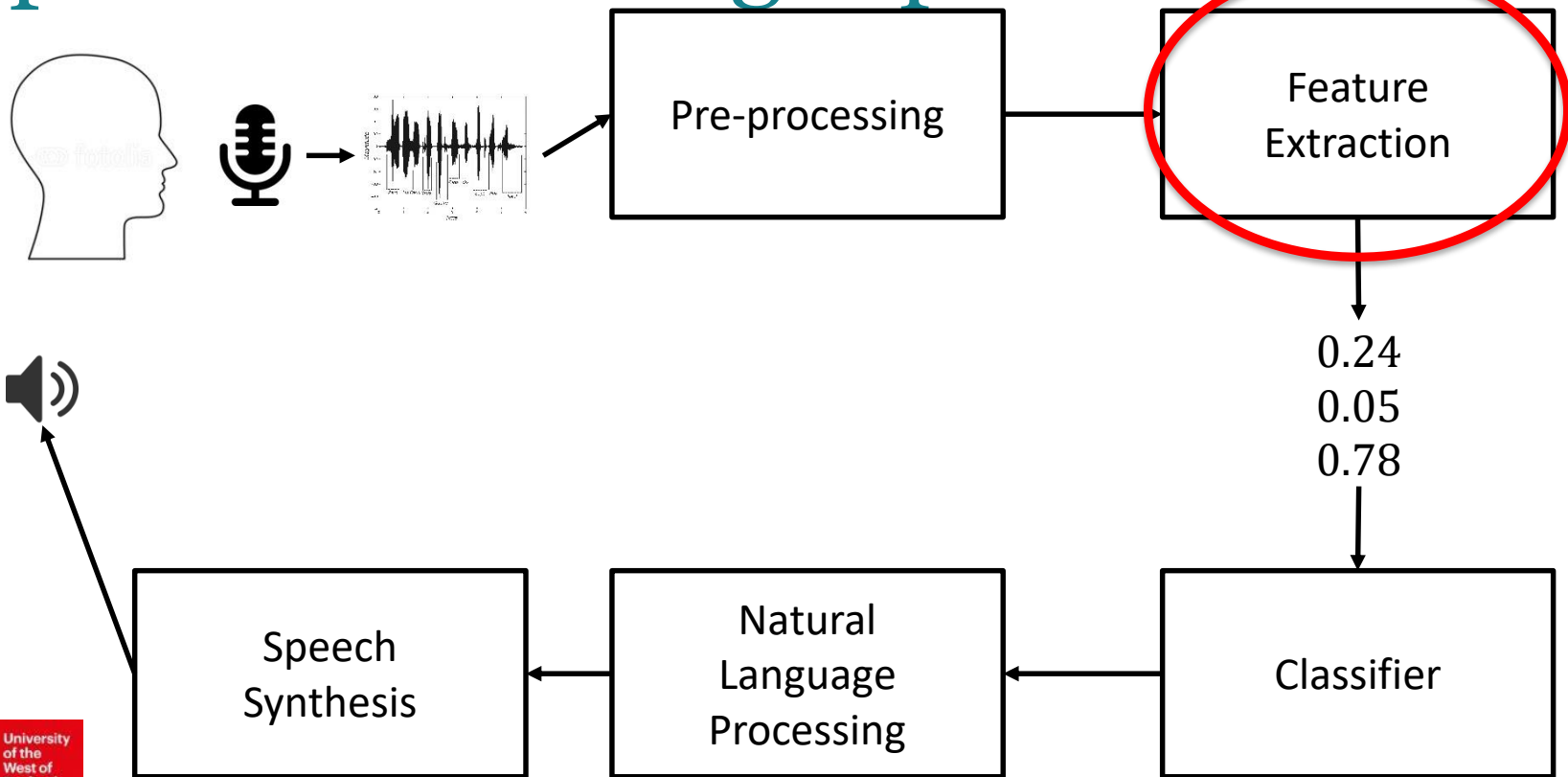


De-noising by
Median filter

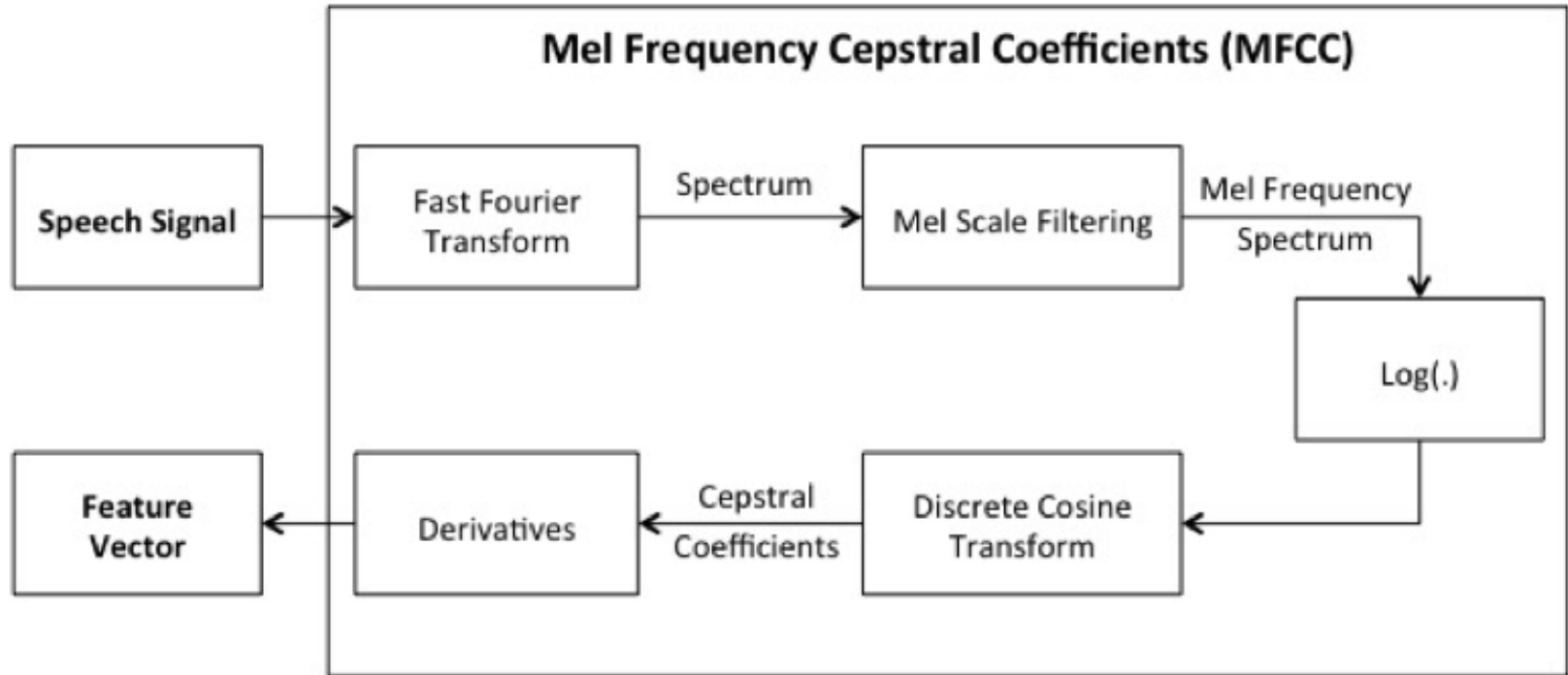
Pre-processing: Denoising

- Reduction of noise in the speech signal
- Typical sources of noise in the signal
 - Microphone
 - Electromagnetic noise
 - Background noise
- Speech recognizers can be trained to ignore microphone noise and electromagnetic noise as they are often static. Background noise is variable and therefore needs to be filtered out as much as possible.
- Frequently used filter techniques
 - Adaptive Wiener Filtering
 - Spectral Subtraction Methods
 - Spectral Restoration (speech enhancement)
 - Harmonic Decomposition
 - Non-Negative Matrix Factorization

Speech Processing Pipeline

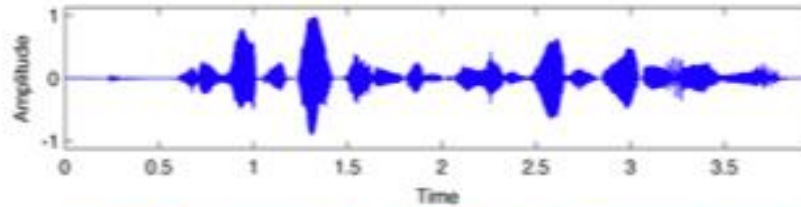


Feature Extraction

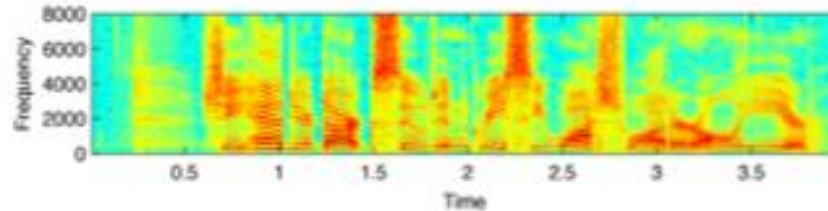


Feature Extraction

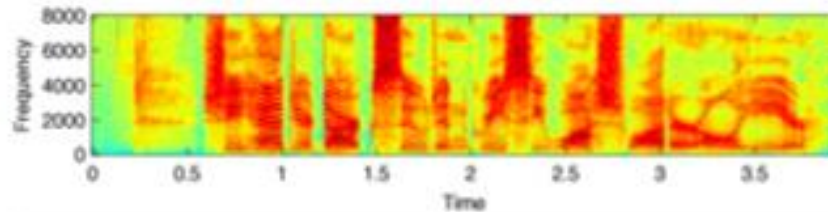
**Time Domain
Waveform**



Spectrogram



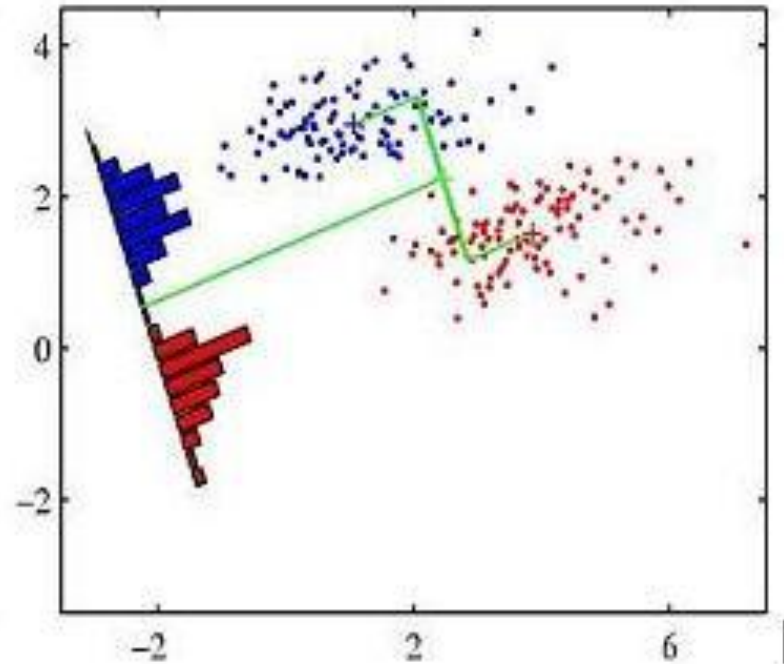
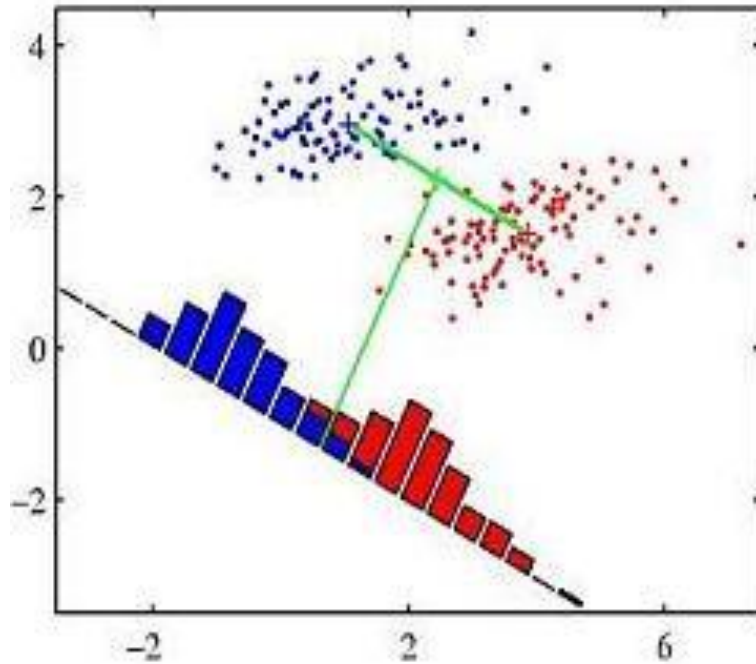
**MFCC
Spectrogram**



Feature Extraction

- The raw speech signal contains additional information that is not relevant to speech recognition, e.g. which speaker is currently speaking
- Feature extraction: Reducing the dimensionality of the speech signal without loss of linguistic information
- Objective: To generate a feature vector that contains only the necessary information to correctly classify the spoken words
- Researchers have proposed different features that emphasize different aspects of the speech signal.
- Well-known suggested features are:
 - Linear Predictive Coding (LPC)
 - Perceptual Linear Predictive Coefficients
 - Mel-Frequency Cepstral Coefficients
 - Linear Prediction Cepstral Coefficients
 - Wavelet Based Features
 - Non-Negative Matrix Factorization features

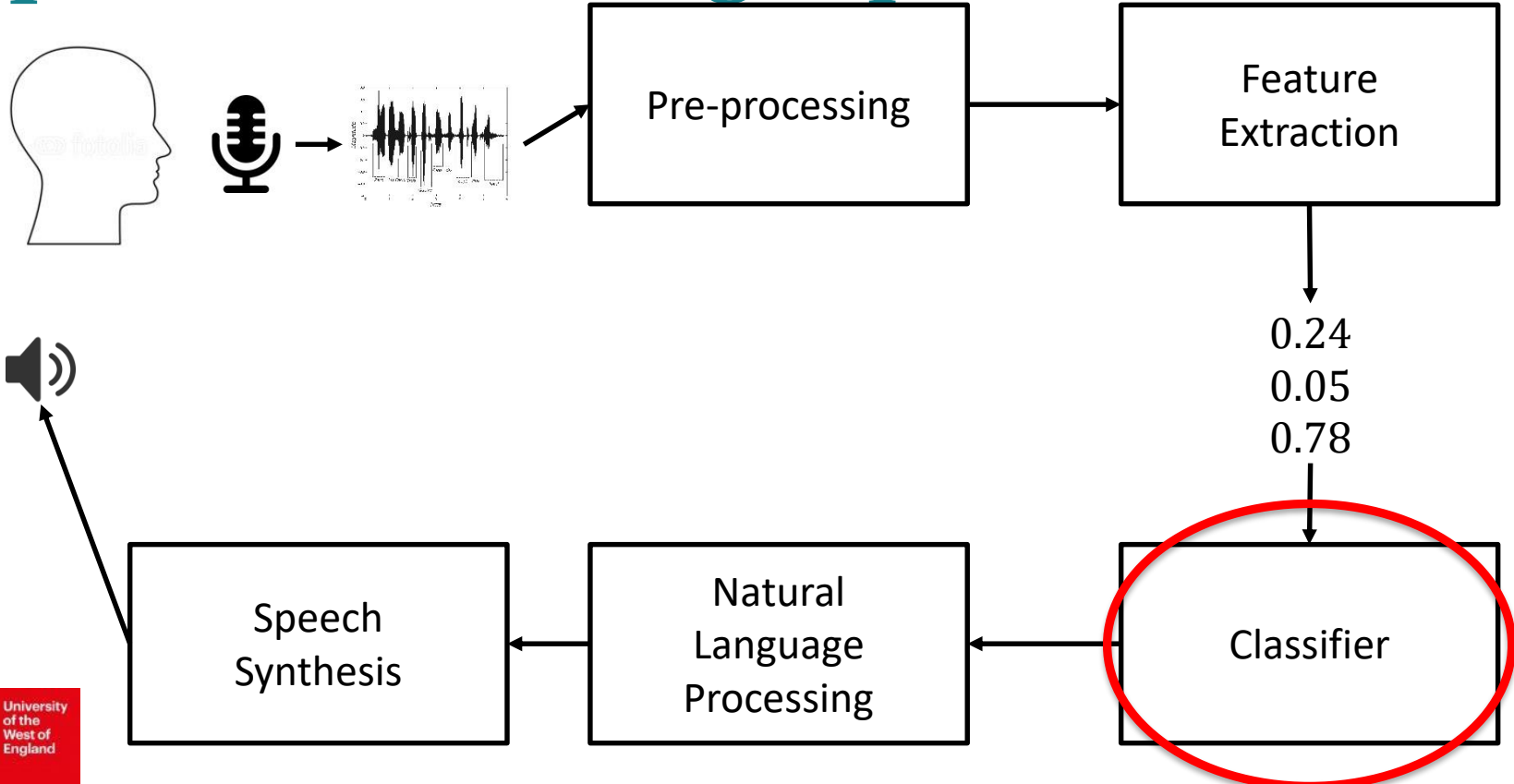
Feature Combination



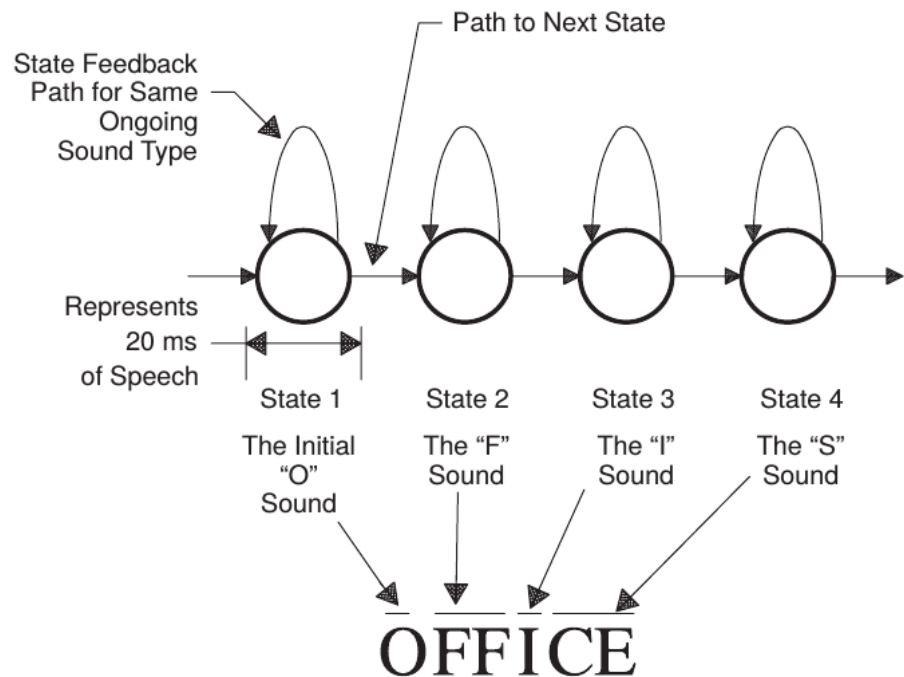
Feature Combination

- Low level features often use quasi-stationary window sizes between 20ms and 30ms. However, it is known that mammals use audio signals with window sizes of about 200ms. Therefore, researchers have begun to use longer window sizes for speech recognition.
- In addition, different features are extracted, stacked (feature stacking), and then reduced again with methods such as the Principal Component Analysis or Linear Discriminant Analysis. Frequently, artificial neural networks are used in this step, which automatically select suitable features with a high information content.

Speech Processing Pipeline



Classifier, Acoustic Modelling



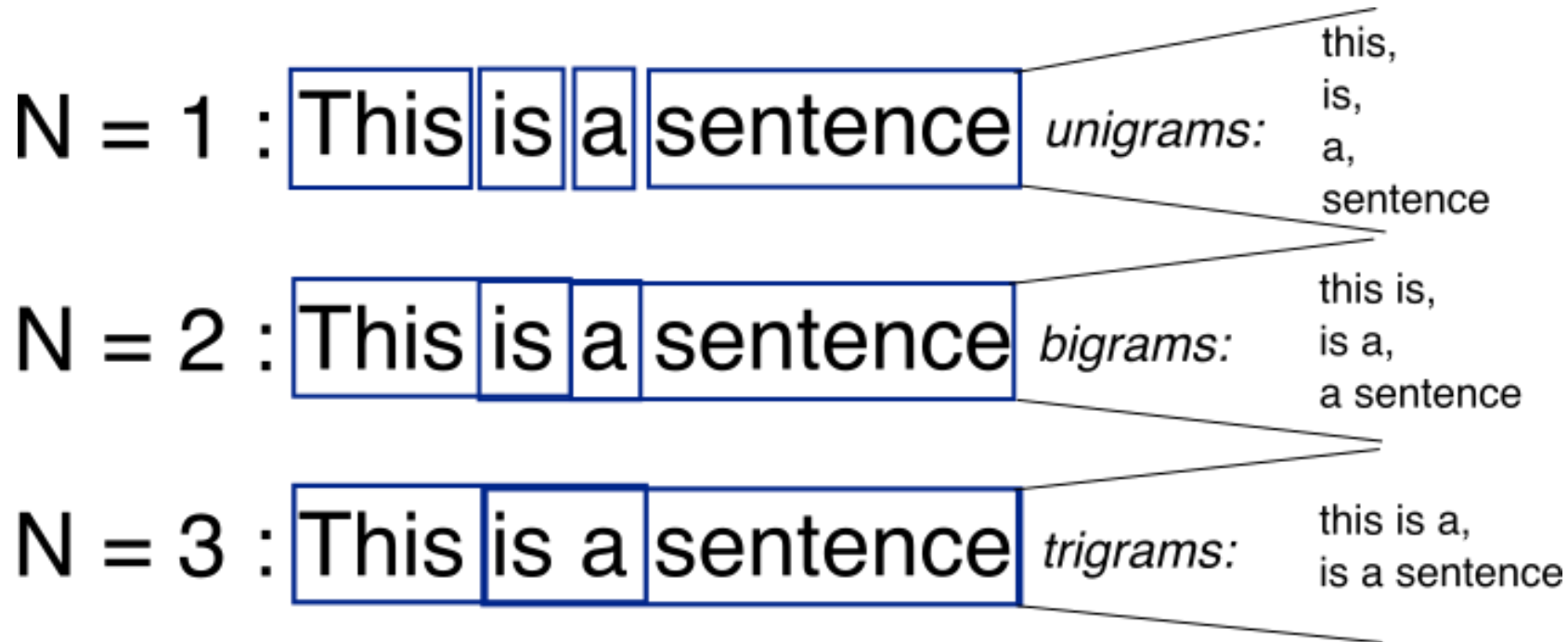
Classifier

- After pre-processing and feature extraction, the classifier has the task of finding the matching word sequence W for a given set of feature vectors X .
- For this step, an acoustic model is first used that calculates the probability of a given feature vector for a particular word. Then the likelihood of the order of words is calculated using a language model.

Acoustic Modelling

- In most speech recognition programs, the acoustic model is implemented using Hidden Markov Models.
- When using HMMs, the question arises as to what should be displayed in the states of the HMMs. Intuitively, one would say in speech recognition that every state should stand for one word. However, since languages contain many words, the corresponding HMM would have too many states. Therefore one uses the phonemes of a language as states.
- A simple acoustic model is e.g. if you take a HMM with three states for each phoneme. In this way, different phonemes can be related to words.

Language Modelling



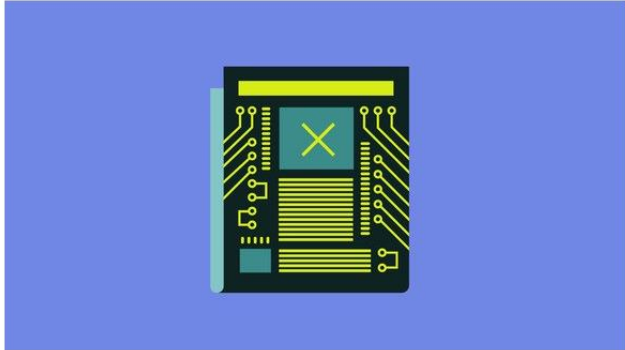
Language Model

- The acoustic model generates a list of word sequences along with their probabilities. With a language model, this list can be additionally evaluated.
- One possible language model is e.g. a phonetic model that calculates how likely it is that two phonemes in a language are spoken one behind the other. In English, for example, it is not very likely that the phonemes [t] and [k] are pronounced directly one after the other.
- Another language model is the "n-gram model". This calculates the probability that n words follow each other. The probabilities can be e.g. calculate from text collections. In practice, 2-gram or 3-gram models are often used. These are called bigrams or trigrams.

Generative Language Models

TOM SIMONITE BUSINESS 02.14.19 12:00 PM

THE AI TEXT GENERATOR THAT'S TOO DANGEROUS TO MAKE PUBLIC



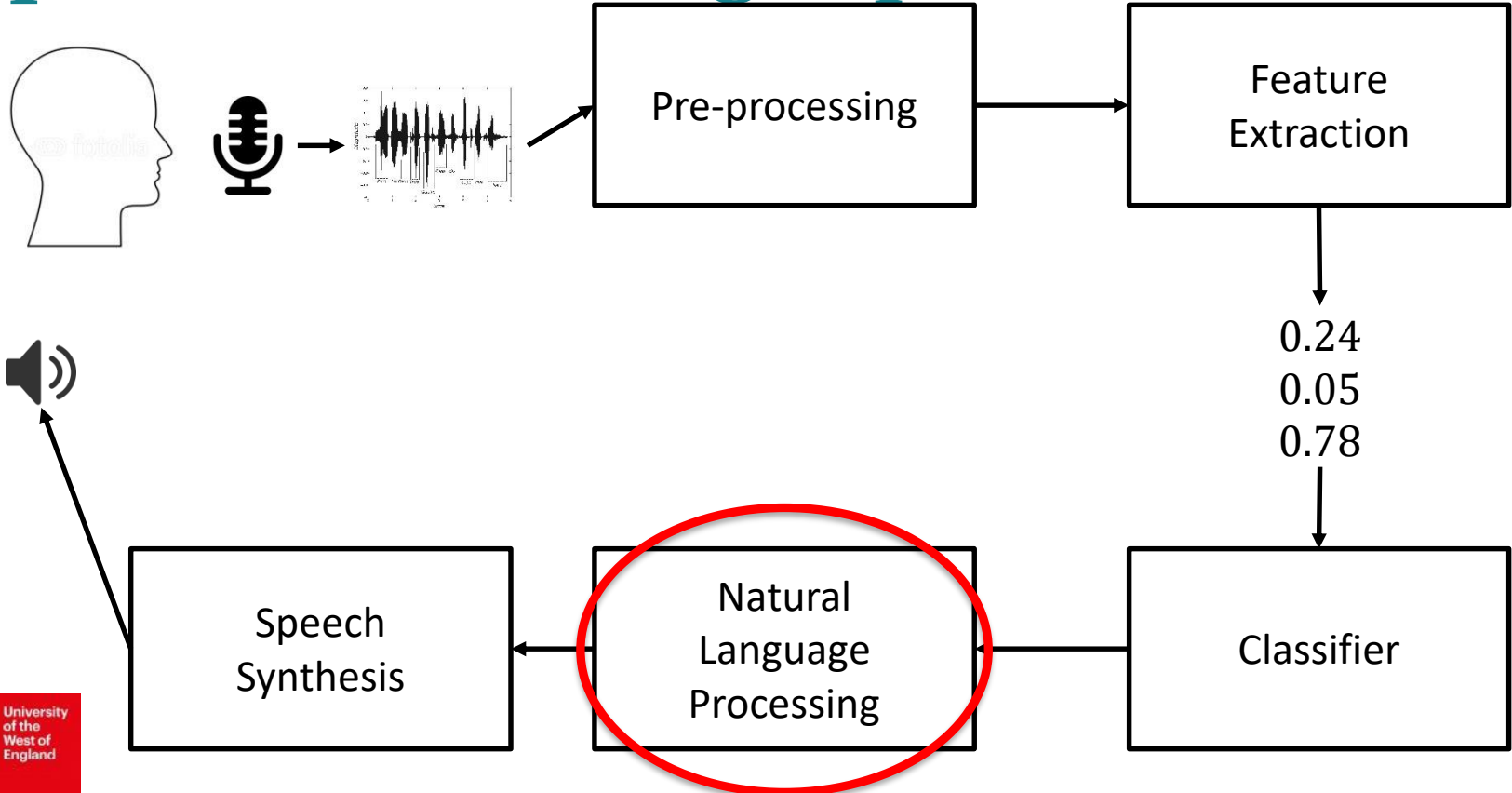
<https://www.wired.com/story/ai-text-generator-too-dangerous-to-make-public/>

<https://blog.openai.com/language-unsupervised/>

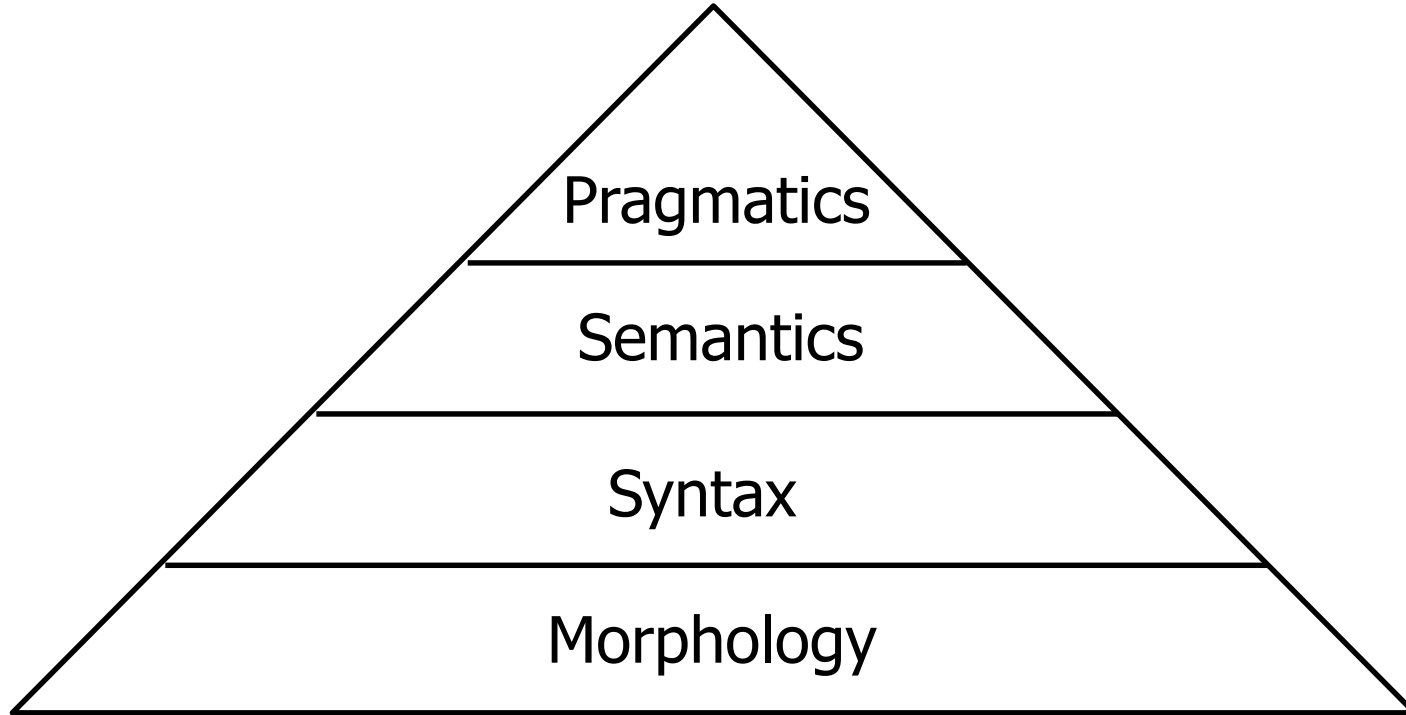
Further Links

- Intro to Google speech recognition
<https://www.youtube.com/watch?v=yxxRAHVtafI>
- Design guidelines for voice response systems
https://downloads.avaya.com/elmodocs2/convsnt/v7_i2/313801_1/310670_1/310670_1.pdf
<http://www.speech.cs.cmu.edu/air/papers/SpInGuidelines/SpInGuidelines.html>
- Youtube channel explaining deep learning (with speech recognition example)
<https://www.youtube.com/channel/UC9OeZkIwhzfv-Cb7fCikLQ>
- Google speech recognition API reference
<https://cloud.google.com/speech-to-text/>
- IBM speech recognition API reference
<https://www.ibm.com/watson/services/speech-to-text/>

Speech Processing Pipeline



NLP Pyramid



NLP Pyramid

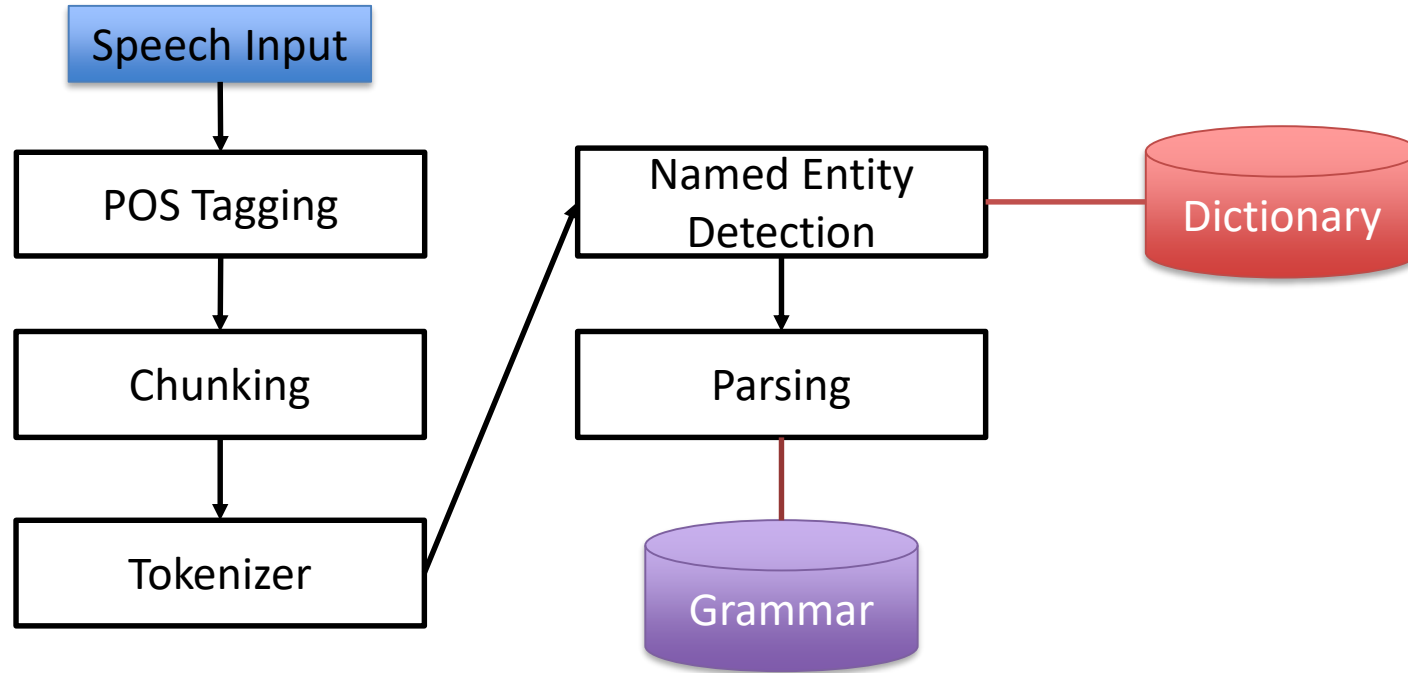
Morphology: Processing at word-level, e.g. detection of word form, how words change depending of context, word inflection, word gender detection, spell checking.

Syntax: Processing of relationship of words within a sentence - how a sentence is constructed. Often done with parsers that use a grammar.

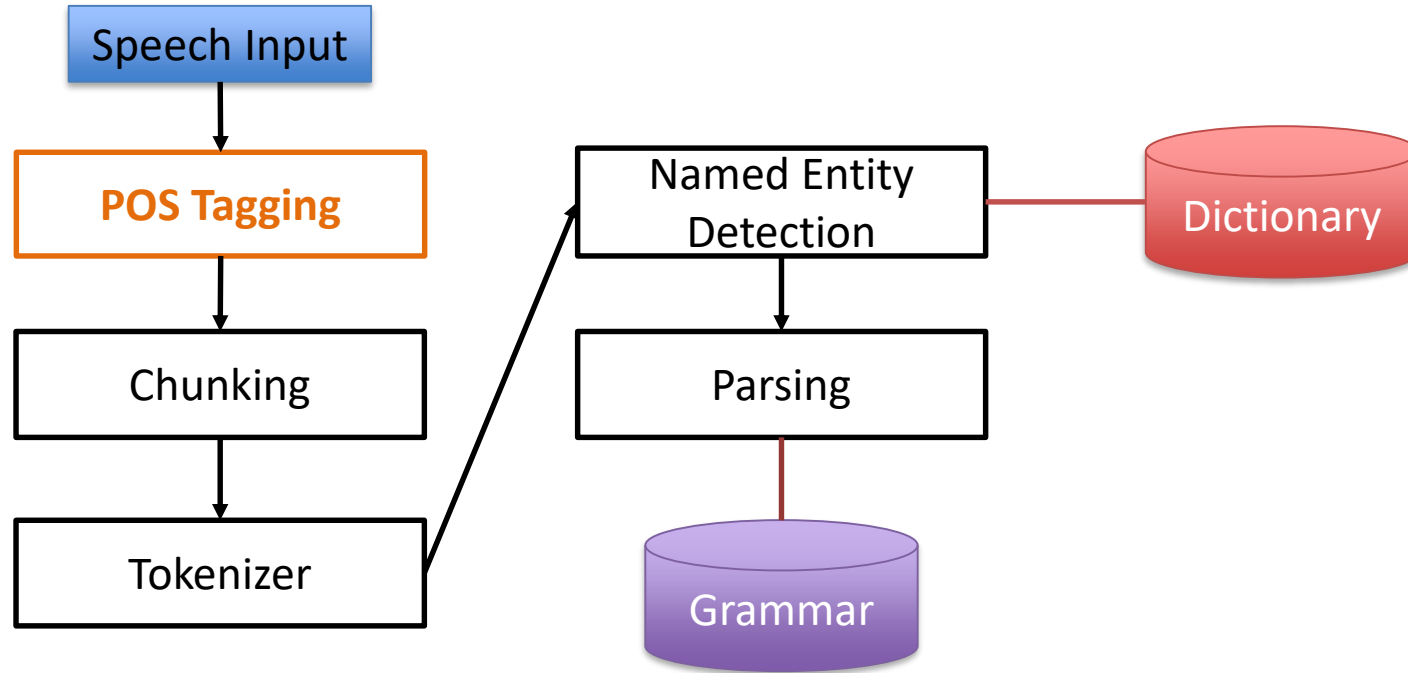
Semantics: Processing of the meaning of a sentence. E.g. named entity recognition, extraction of speech acts (i.e. what did the speaker want to express with the sentence, e.g. a question or a request), relationship extraction.

Pragmatics: Understanding of the text as a whole. For example, topic modelling, text summarisation, question answering, anaphora resolution.

Syntactic NLP

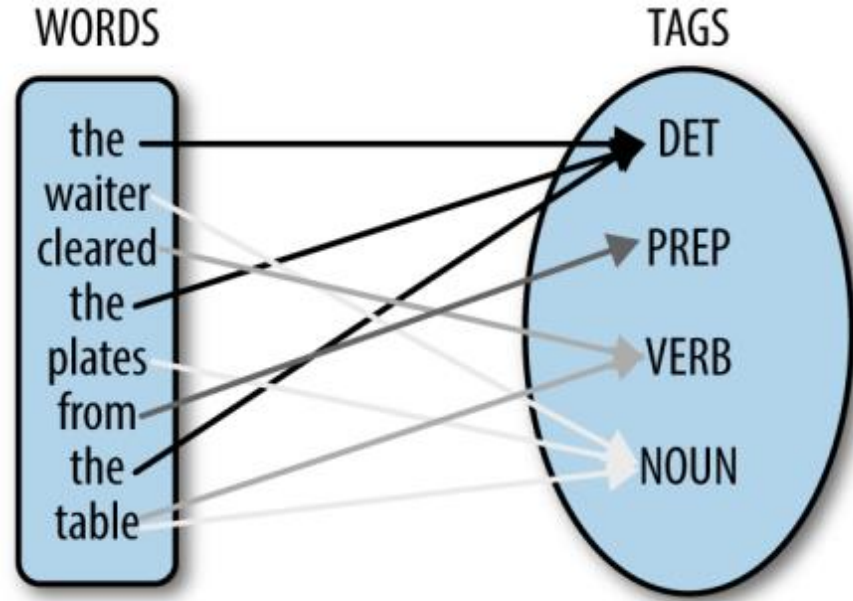


Syntactic NLP



Part of Speech Tagging

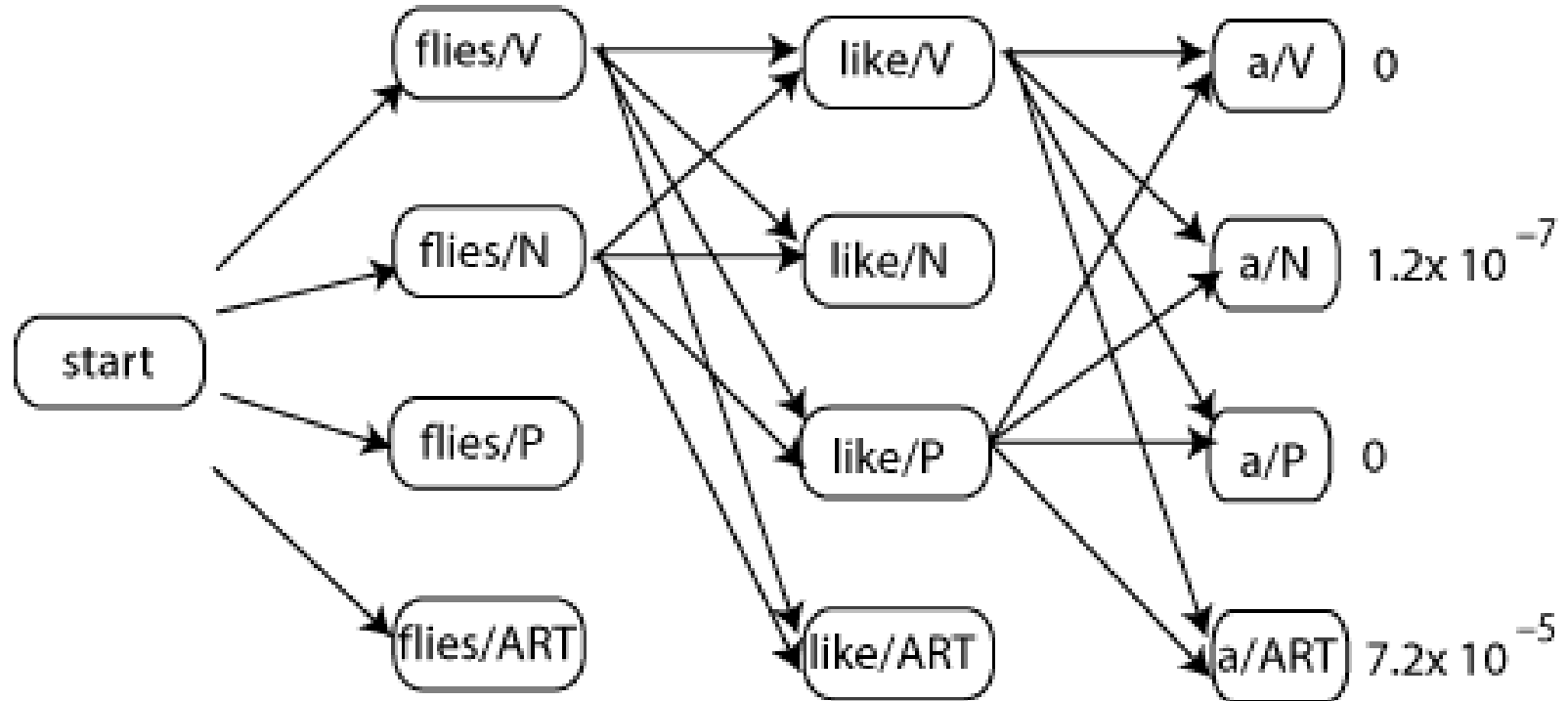
- Detection of the part of speech of a word (e.g. Adjective, noun, verb)
- Sometimes called “grammatical tagging” or “word category disambiguation”
- POS Tagging is the basic input for most NLP systems
- Can be implemented with rule-based and statistical systems
- Nowadays mostly statistical processing



Most commonly used POS symbols:

<https://sites.google.com/site/partofspeechhelp/>

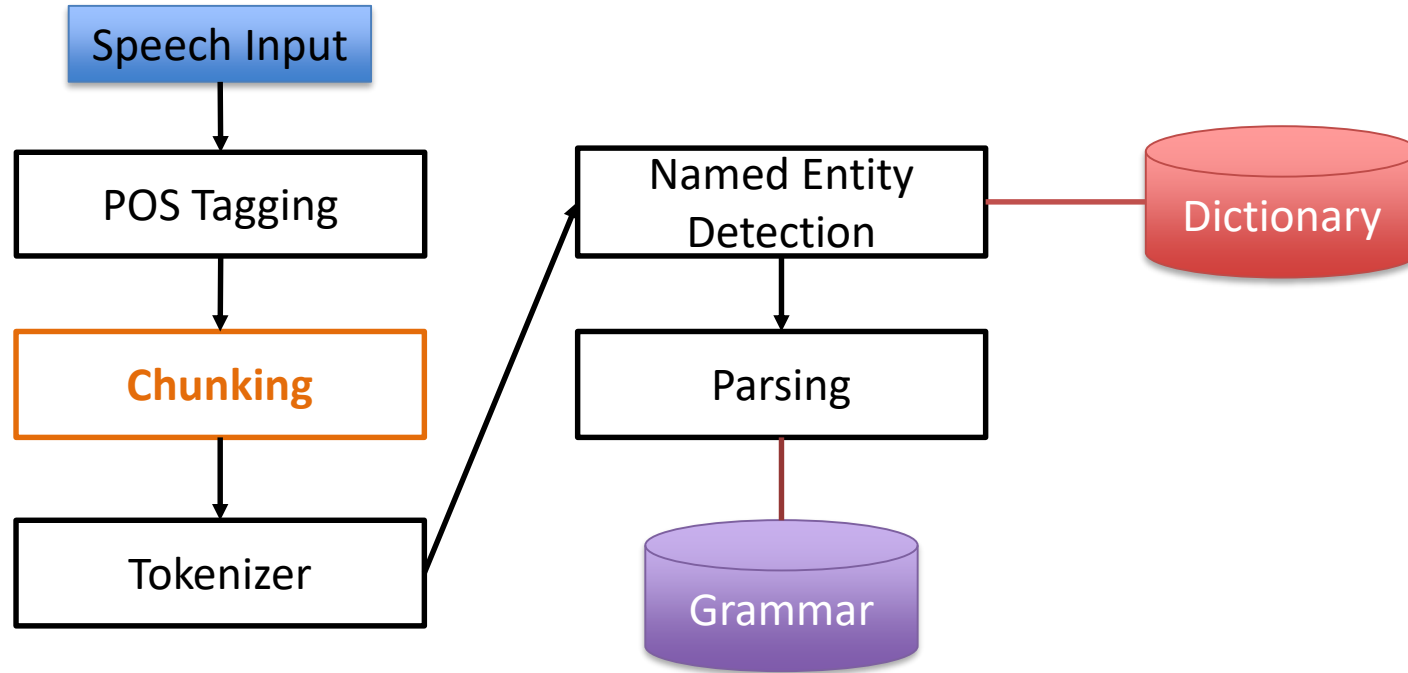
Part of Speech Tagging



Part of Speech Tagging

- <https://parts-of-speech.info/>
- <http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>

Syntactic NLP



Chunking

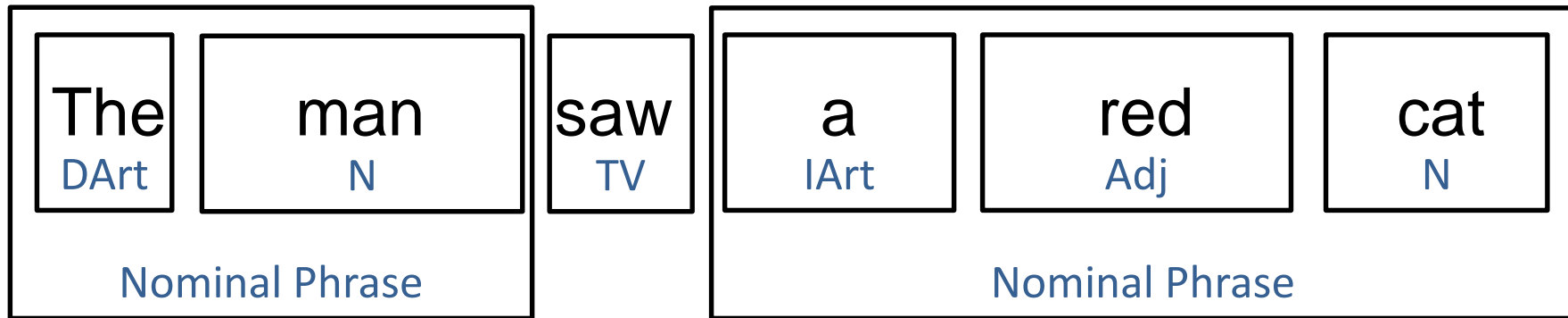
- Identify sentence parts
- Identification of
 - Proper names
 - Descriptions
 - Phrases
 - Sentence ends
- Easier than full sentence parsing
- Sometimes called “shallow parsing”

Chunking

In chunking, parts of sentences are identified based on the word types of the words. Sentence parts are for example noun phrases or verb phrases.

Chunking is used to find keywords or proper names or to identify phrases in order to establish a relationship between them. Chunking is easier to use than parsing because it is also applicable to incomplete sentences. Therefore, chunking is often called "Shallow Parsing".

Chunking Example



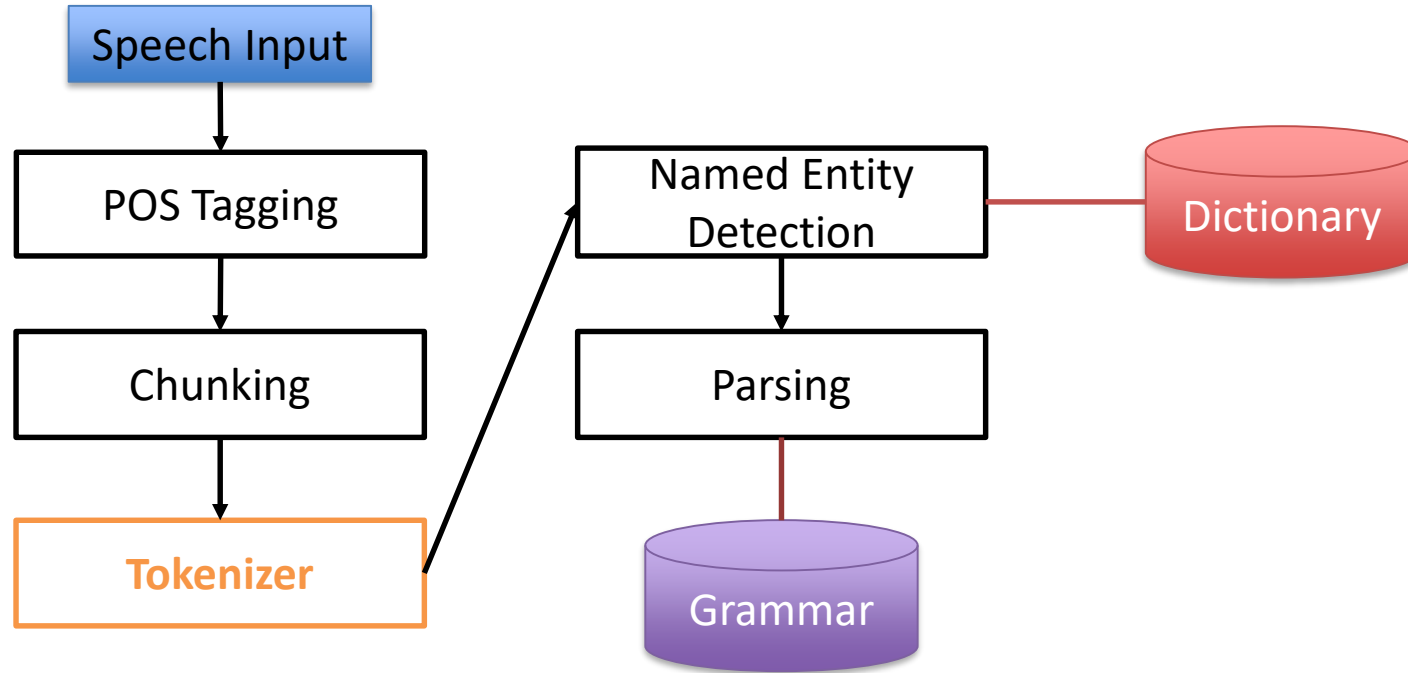
Chunking

- Simple chunking can be implemented with a POS tagger and regular expressions
- Example text
The/DT man/NN made/V another/DT sharp/JJ dive/NN
trade/NN and/Conj hit/V the/DT green/JJ wall/NN
with/Conj his/Pron old/JJ car/NN
- Regular expression: <DT>?<JJ.*>*<NN.*>+
- Which sentence parts do you get?

Chunking

- State of the Art in NP Tagging at
[https://aclweb.org/aclwiki/NP_Chunking_\(State_of_the_art\)](https://aclweb.org/aclwiki/NP_Chunking_(State_of_the_art))
- Chunking demonstration at
https://cogcomp.org/page/demo_view/ShallowParse

Syntactic NLP

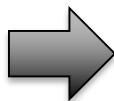


Tokenizer

- Converting a sequence of characters into a sequence of tokens

Example:

The quick brown fox
jumps over the lazy dog

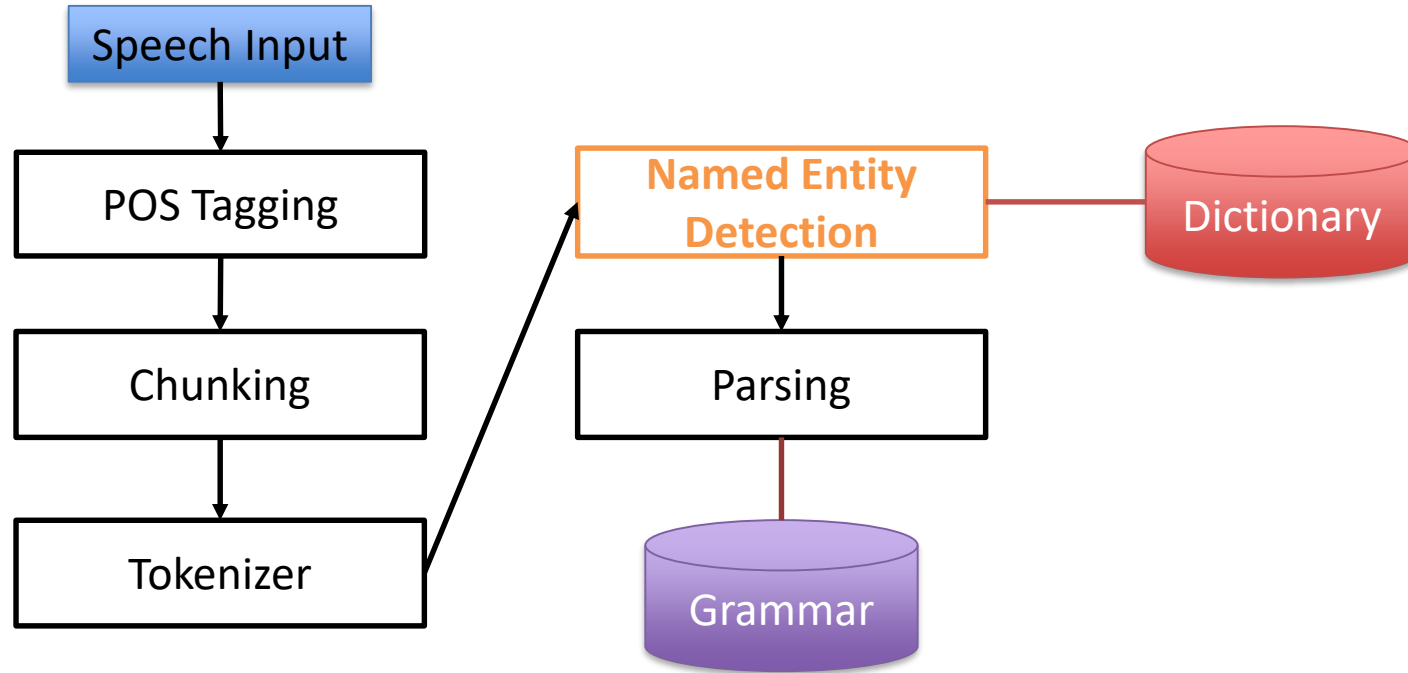


```
<sentence>  
  <word>The</word>  
  <word>quick</word>  
  <word>brown</word>  
  <word>fox</word>  
  <word>jumps</word>  
  <word>over</word>  
  <word>the</word>  
  <word>lazy</word>  
  <word>dog</word>  
</sentence>
```

Tokenizer

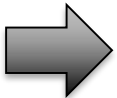
- Apache OpenNLP Tools Tokenizer
<https://opennlp.apache.org/docs/1.9.1/manual/opennlp.html#tools.tokenizer.cmdline>

Syntactic NLP



Named Entity Detection

Jim bought 300 shares
of Acme Corp. in 2006.



[Jim]_{Person} bought 300 shares of
[Acme Corp.]_{Organization} in [2006]_{Time}.

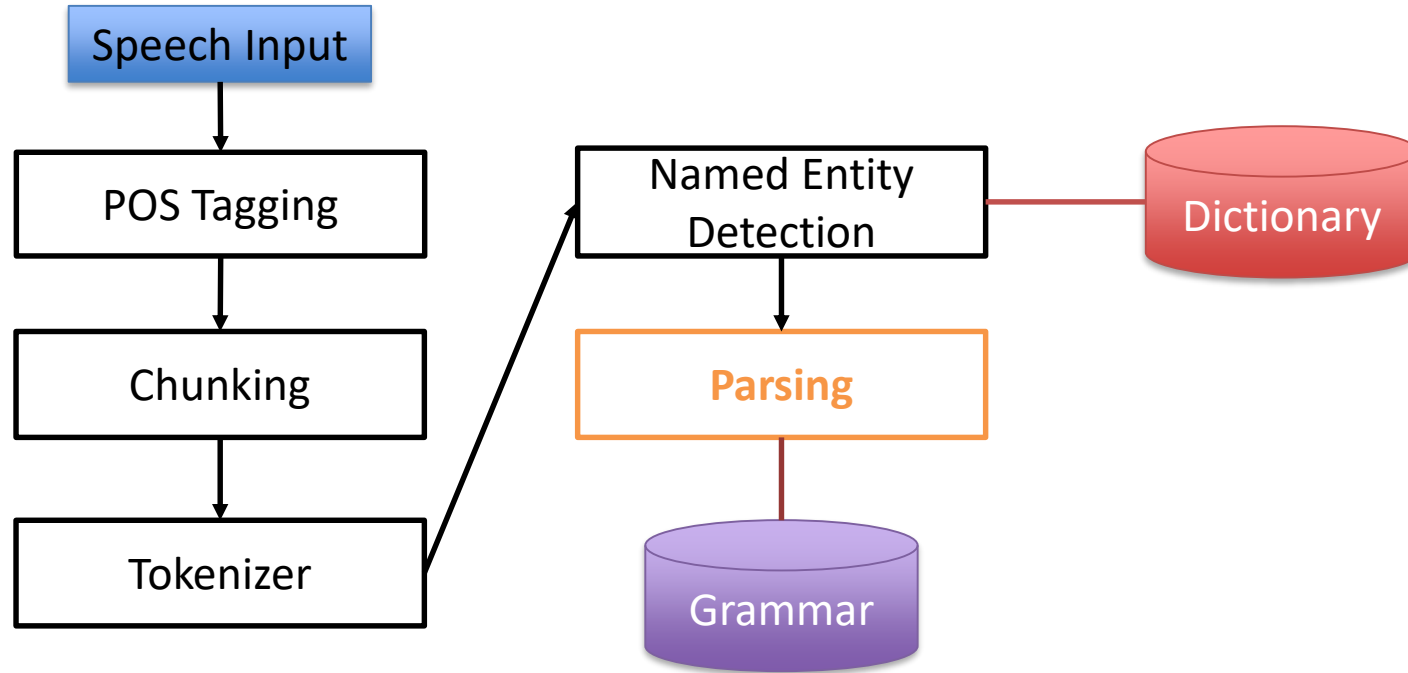
Named Entity Detection

- Named entity detection (NED) is also called named entity recognition sometimes. NED seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as person names, organisations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

Named Entity Detection

- Named Entity Finder in Apache OpenNLP
<https://opennlp.apache.org/docs/1.9.1/manual/opennlp.html#tools.namfind.recognition>
- Spacy NLP in Python has a Named Entity Detection
<https://spacy.io>

Syntactic NLP



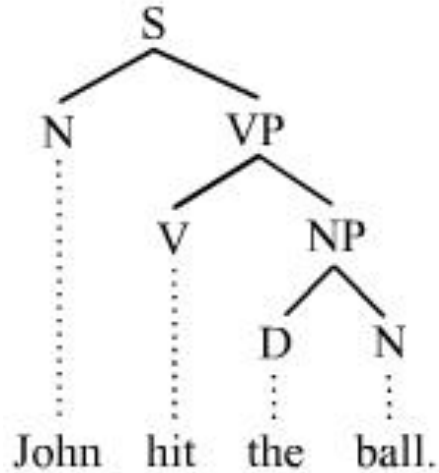
Parsing

- Parser: Program that converts an input into a format suitable for further processing
- In the case of speech, parsing refers to syntax analysis
- Parsing of written or spoken language
- Transfer of language into logical expressions
- Finding contexts between phrases

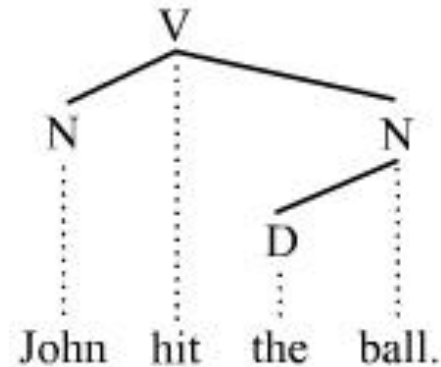
Parsing Technology

- Parsing algorithms
 - Top-down
 - Bottom-up
- Context free vs. Context-sensitive grammars
- Types of grammars
 - Constituent grammars
 - Dependency grammars
 - Unification grammars
 - Categorical grammars

Parsing Technology

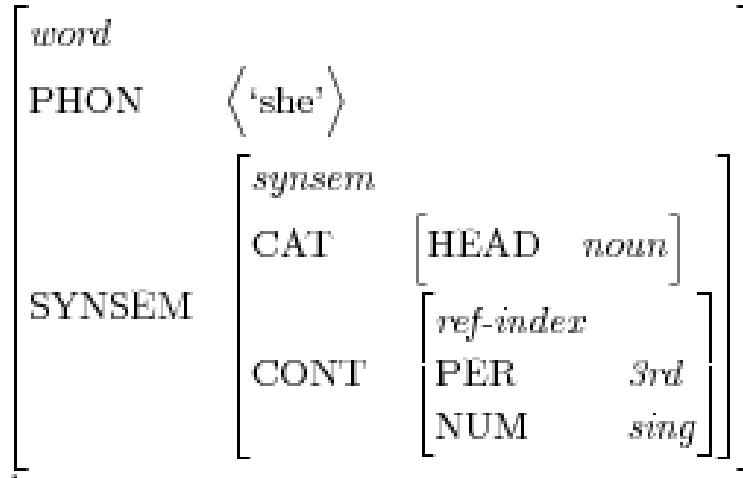


Constituency-based
parse tree

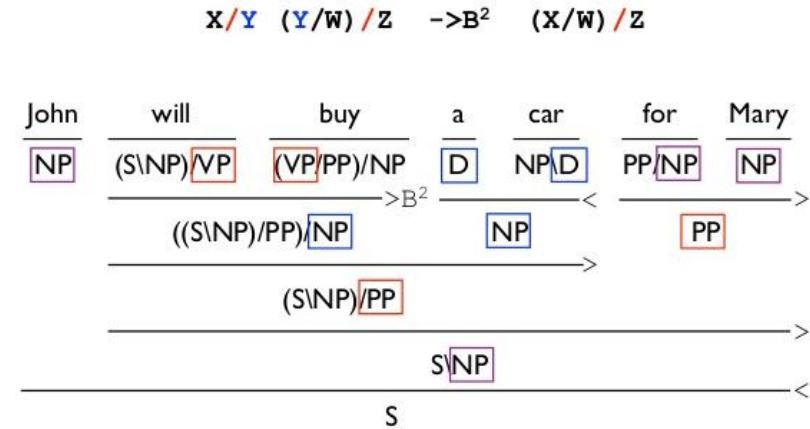


Dependency-based
parse tree

Parsing Technology



Unification-based
parse



Categorial parse

Parsing Links

- Stanford Parser, statistical Parser
<http://nlp.stanford.edu/software/lex-parser.shtml>
- OpenCCG, write your own categorial grammars in Java
<http://openccg.sourceforge.net/>
- Google Syntaxnet, Parsey McParseface
<https://ai.googleblog.com/2017/03/an-upgrade-to-syntaxnet-new-models-and.html>
https://github.com/tensorflow/models/tree/master/research/syntaxnet/syntaxnet/models/parsey_mcparseface

Self check

How does speech recognition work?

What does a chunker do?

What are two typical parsing methods?