Introduction to Natural Language Processing

Matthieu Labeau

matthieu.labeau@telecom-paris.fr

Language Processing: goals

Interdisciplinary field, whose goal is to get computers to perform useful tasks [..] like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech.

Speech and Language Processing (Jurafsky & Manning) (Chapter I)

Language Processing: goals

Interdisciplinary field, whose goal is to get computers to perform useful tasks [..] like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech.

Speech and Language Processing (Jurafsky & Manning) (Chapter I)

Applications ?

From the same source:example of HAL 9000, in *2001: A Space Odyssey*

$\rightarrow$ Conversational agent - what does HAL imply ?

Language Processing: goals

Interdisciplinary field, whose goal is to get computers to perform useful tasks [..] like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech.

Speech and Language Processing (Jurafsky & Manning) (Chapter I)

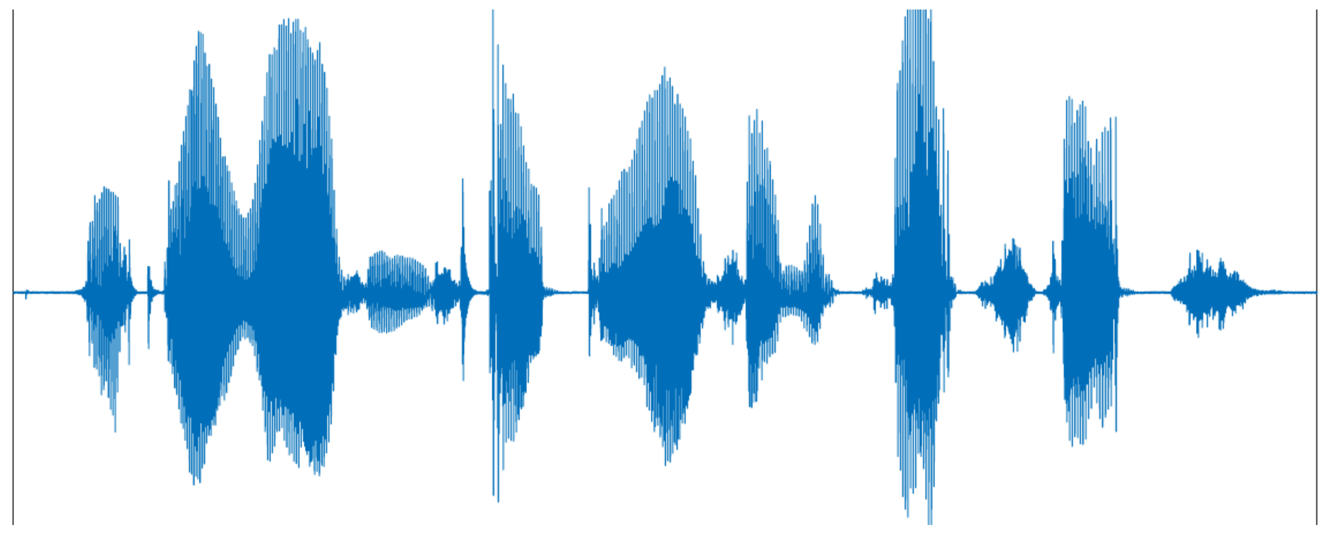
Applications ?

From the same source:example of HAL 9000, in *2001: A Space Odyssey*

$\rightarrow$ Conversational agent - what does HAL imply ?

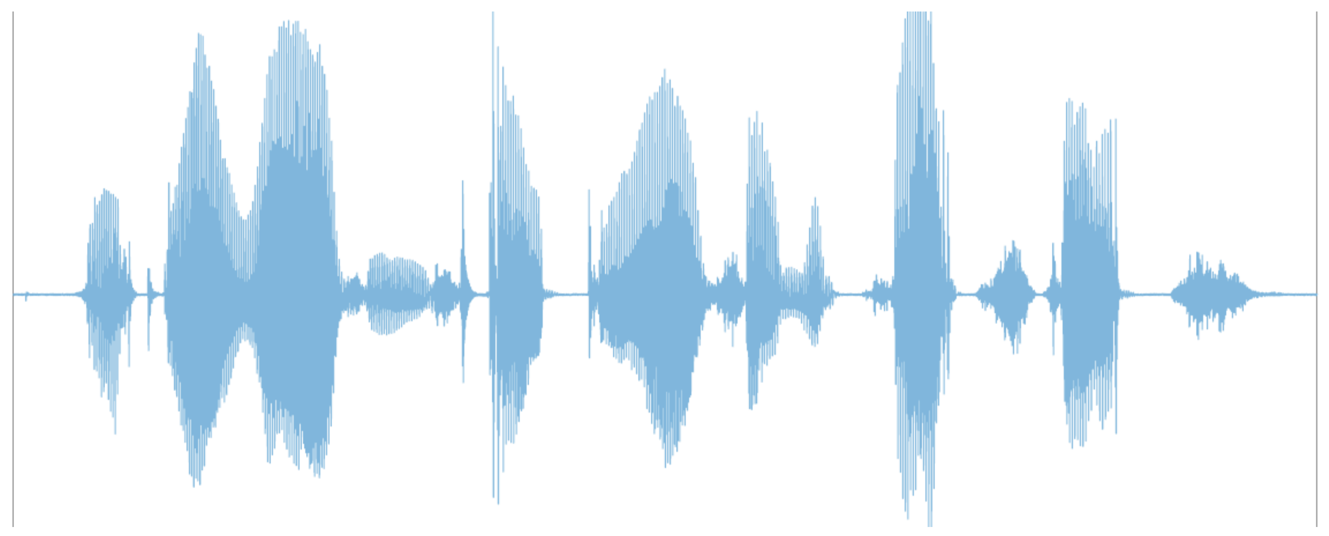
- Language input: *speech recognition*, *language understanding*
- Language output: *dialogue planning*, *speech synthesis*
- Information *retrieval*, *extraction*, and doing *inference* from it

Analyzing language: tasks involved



Signal

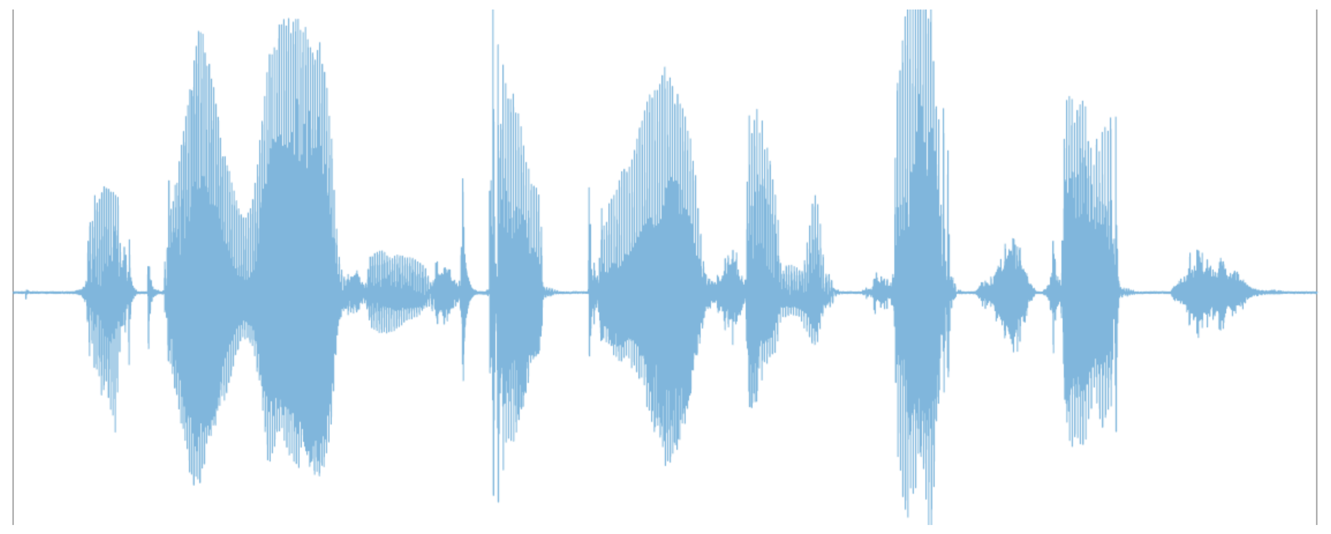
Analyzing language: tasks involved



$$\begin{tikzcd}\text{Signal} \arrow[d] \\ \text{Phonemes}\end{tikzcd}$$

ɪt pɪktʃəz eɪ kənstɹɪktə bəʊə dɪd͡ʒestɪŋ ən elɪfənt

Analyzing language: tasks involved

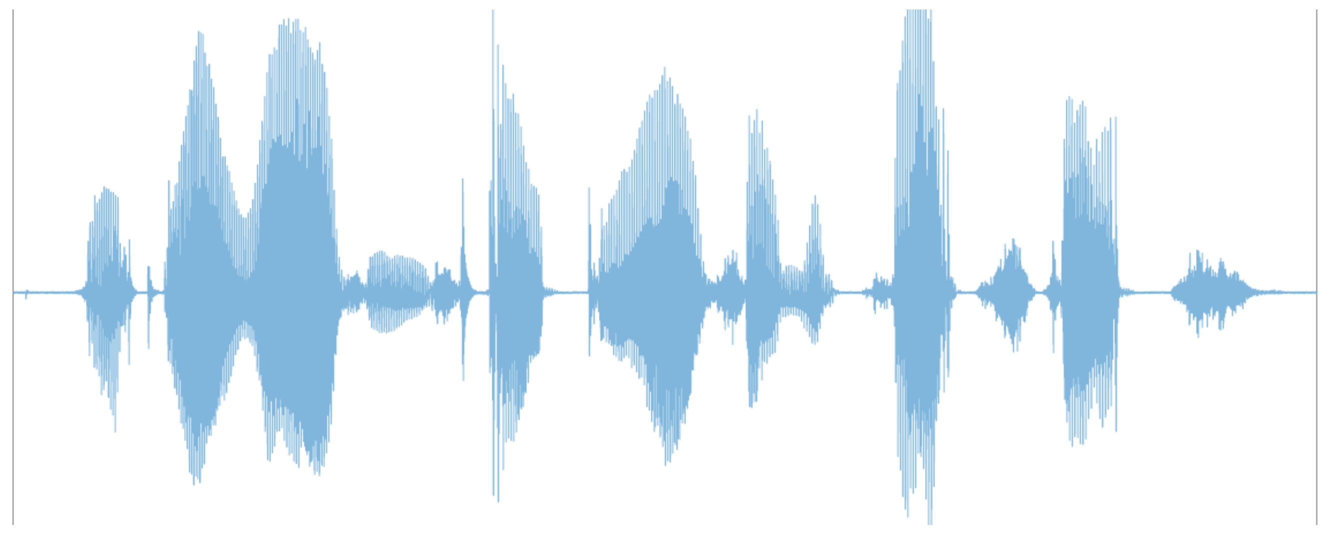


ɪt pɪktʃəz eɪ kənstɹɪktə bəʊə dɪd͡ʒestɪŋ ən elɪfənt

It pictures a constrictor boa digesting an elephant

$$\begin{tikzcd}
\text{Signal} \arrow[d] \\
\text{Phonemes} \arrow[d] \\
\text{Text}
\end{tikzcd}$$

Analyzing language: tasks involved



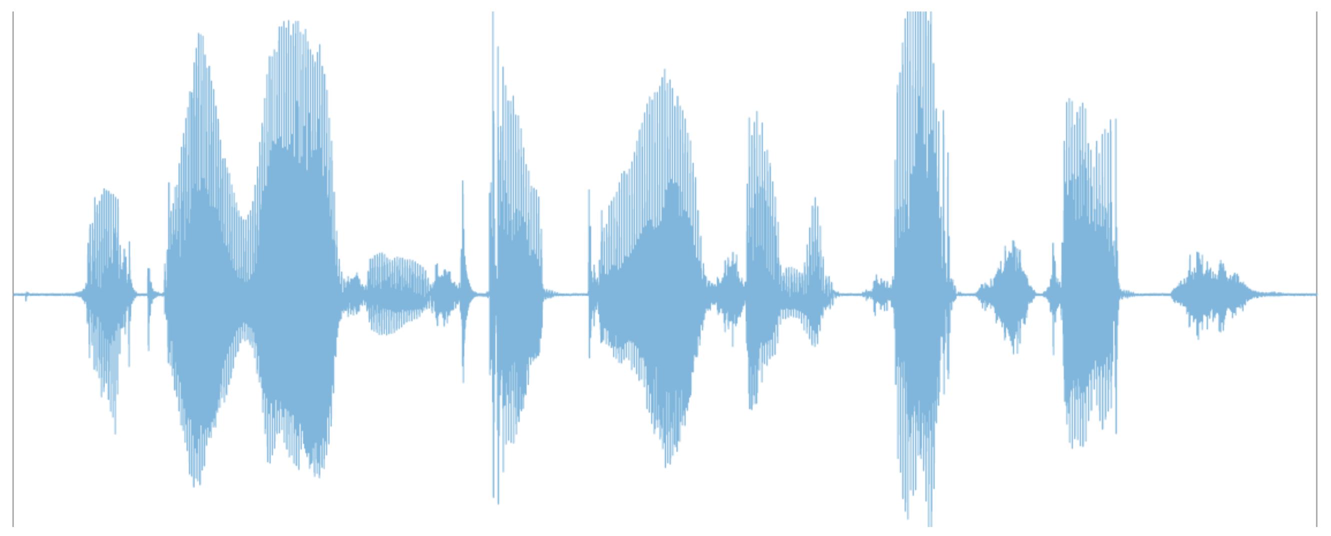
ɪt pɪktʃəz eɪ kənstɹɪktə bəʊə dɪd□ʒestɪŋ ən elɪfənt

It pictures a constrictor boa digesting an elephant

PRP VBZ DT NN NN VBG DT NN

$$\begin{tikzcd}
\text{Signal} \arrow[d] \\
\text{Phonemes} \arrow[d] \\
\text{Text} \arrow[d] \\
\text{POS Tags}
\end{tikzcd}$$

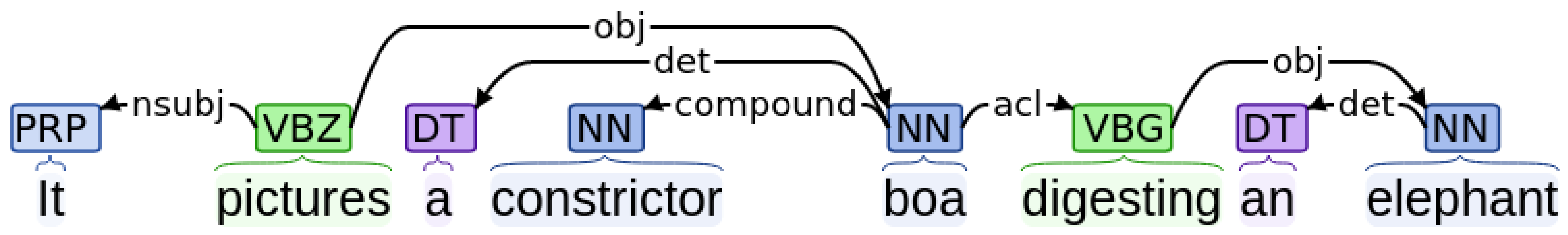
Analyzing language: tasks involved



ɪt pɪktʃəz eɪ kənstɹɪktə bəʊə dɪd□ʒestɪŋ ən elɪfənt

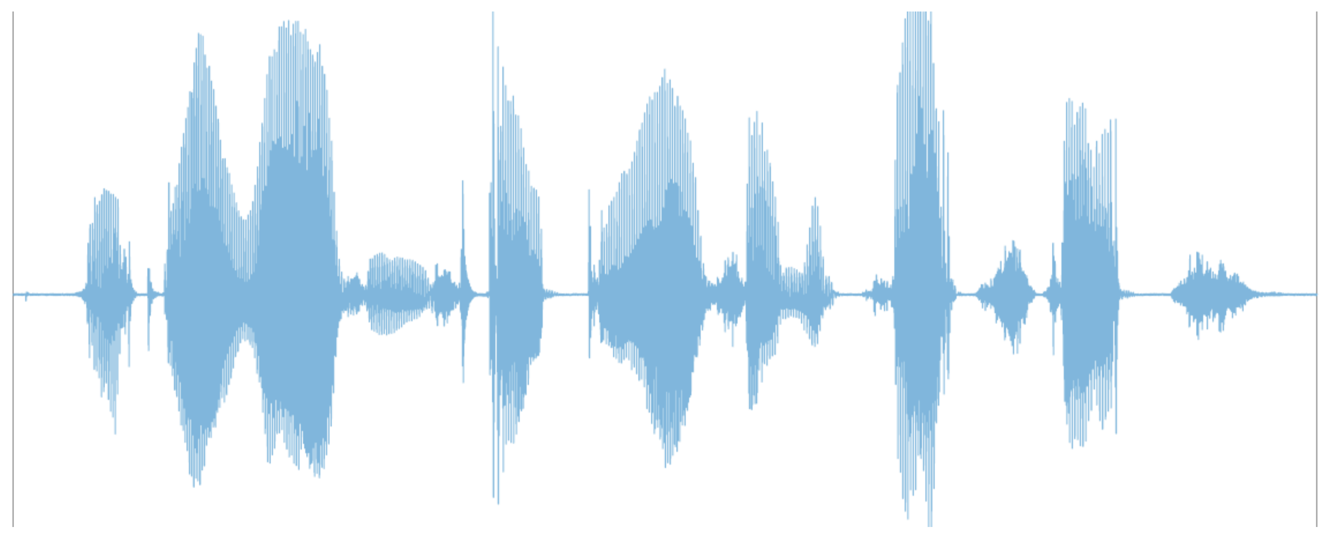
It pictures a constrictor boa digesting an elephant

PRP VBZ DT NN NN VBG DT NN



$$\begin{tikzcd}
\text{Signal} \arrow[d] \\
\text{Phonemes} \arrow[d] \\
\text{Text} \arrow[d] \\
\text{POS Tags} \arrow[d] \\
\text{Dependency Tree}
\end{tikzcd}$$

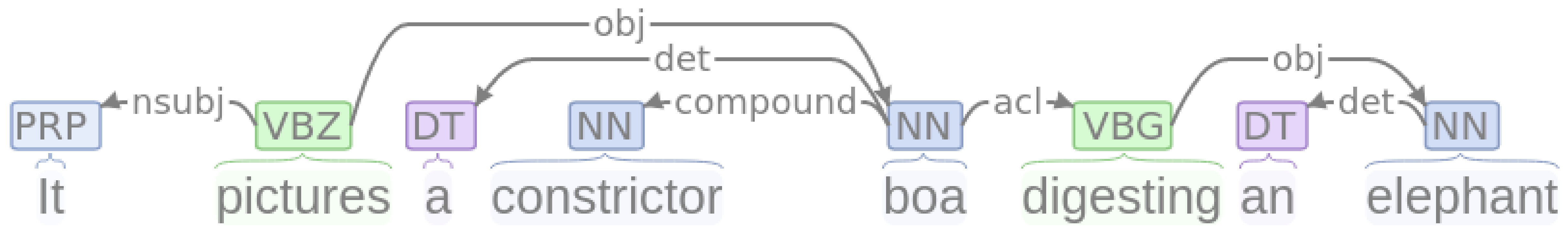
Analyzing language: tasks involved



ɪt pɪktʃəz eɪ kənstɹɪktə bəʊə dɪd□ʒestɪŋ ən elɪfənt

It pictures a constrictor boa digesting an elephant

PRP VBZ DT NN NN VBG DT NN



```
(p / picture-01
   :ARG0 (i / it)
   :ARG1 (b2 / boa
            :mod (c / constrictor)
            :ARG0-of (d / digest-01
                         :ARG1 (e / elephant))))
```

$$\begin{tikzcd}
\text{Signal} \arrow[d] \\
\text{Phonemes} \arrow[d] \\
\text{Text} \arrow[d] \\
\text{POS Tags} \arrow[d] \\
\text{Dependency Tree} \arrow[d] \\
\text{Abstract Meaning Representation}
\end{tikzcd}$$

Different kinds of linguistic knowledge

To perform those tasks, what do we need ?

Different kinds of linguistic knowledge

To perform those tasks, what do we need ?

To enable completely understanding a statement, we need to work at different *levels*:

Different kinds of linguistic knowledge

To perform those tasks, what do we need ?

To enable completely understanding a statement, we need to work at different *levels*:

Segmentation

- Segment text into lexical units (words)
- Not trivial: many possible uses for any punctuation symbol
- No typographic normalization; new uses (emoticons)

Different kinds of linguistic knowledge

To perform those tasks, what do we need ?

To enable completely understanding a statement, we need to work at different *levels*:

Segmentation

- Segment text into lexical units (words)
- Not trivial: many possible uses for any punctuation symbol
- No typographic normalization; new uses (emoticons)

$\rightarrow$ This is usually called **tokenization**

- Pieces are called *tokens*: not necessarily words anymore
- More on that later !

Different kinds of linguistic knowledge

Lexical treatment

- Identify words (*string* to *linguistic unit*) and their properties

Different kinds of linguistic knowledge

Lexical treatment

- Identify words (*string* to *linguistic unit*) and their properties
- Normalizing text and dealing with new words:
  $\rightarrow$ With the help of **morphology**
  $\rightarrow$ How words are built from *morphemes*

Different kinds of linguistic knowledge

Lexical treatment

- Identify words (*string* to *linguistic unit*) and their properties
- Normalizing text and dealing with new words:
  $\rightarrow$ With the help of **morphology**
  $\rightarrow$ How words are built from *morphemes*
  - *Inflectional morphology*
    (dog $\rightarrow$ dogs, play $\rightarrow$ played)
    Variation: same word, but modifies tense, number, ...

Different kinds of linguistic knowledge

Lexical treatment

- Identify words (*string* to *linguistic unit*) and their properties
- Normalizing text and dealing with new words:
  $\rightarrow$ With the help of **morphology**
  $\rightarrow$ How words are built from *morphemes*
  - *Inflectional morphology*
    (dog $\rightarrow$ dogs, play $\rightarrow$ played)
    Variation: same word, but modifies tense, number, ...
  - *Derivational* morphology
    (*happy* $\rightarrow$ *happiness*, *teach* $\rightarrow$ *teacher*)
    Formation: changes meaning, grammatical category

Different kinds of linguistic knowledges

Structural knowledge to assemble words: Syntax

- Constraints to obtain grammatically correct sentences: how words are organized
  $\rightarrow$ Validity in *position* and *agreement*

Different kinds of linguistic knowledges

Structural knowledge to assemble words: Syntax

- Constraints to obtain grammatically correct sentences: how words are organized
  $\rightarrow$ Validity in *position* and *agreement*
- Can be encoded in various ways (*constituency trees, dependency trees*) with their own pros and cons

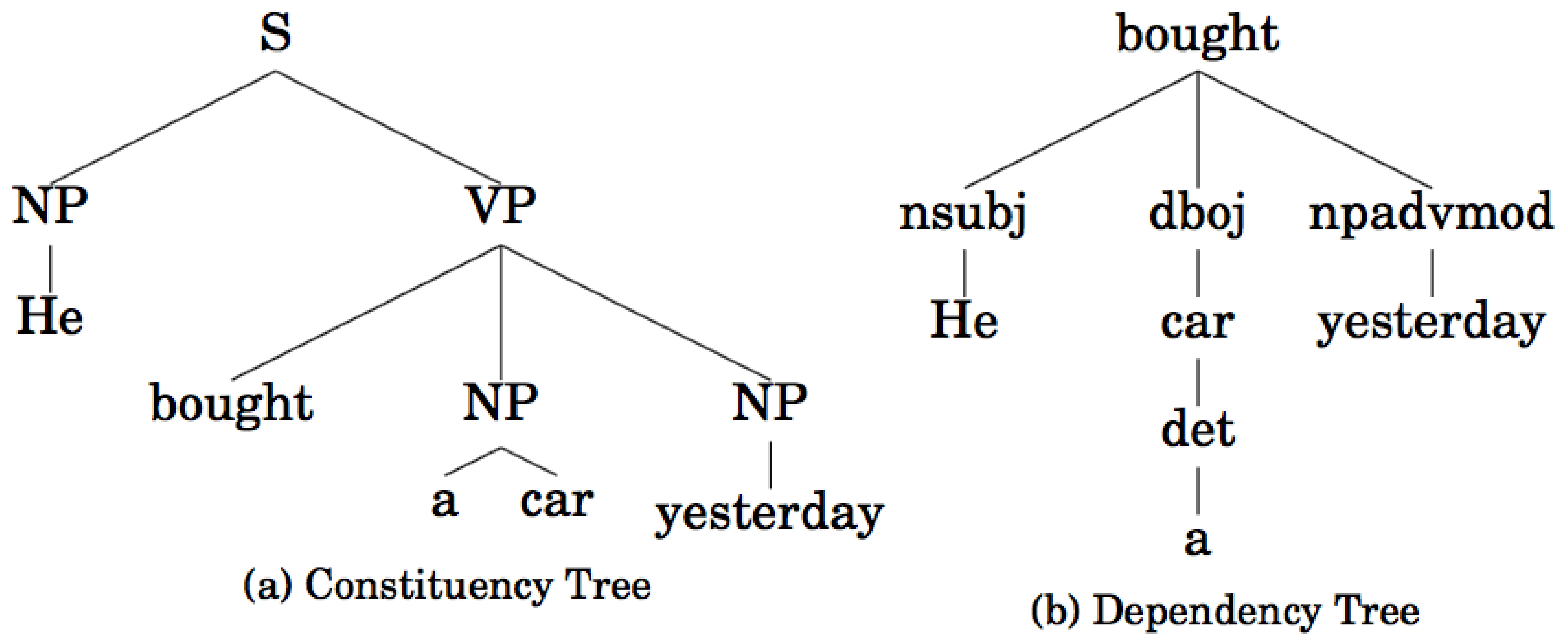


Fig. 3: Examples of the results of constituency and dependency parsing.

Different kinds of linguistic knowledges

Semantics

- How words and sentences carry a meaning
- Lexical semantics: meaning relationships between words (*synonymy, antonymy, hypernymy*)

Different kinds of linguistic knowledges

Semantics

- How words and sentences carry a meaning
- Lexical semantics: meaning relationships between words (*synonymy, antonymy, hypernymy*)
- **Compositionality**
  - Dealing with idioms: *kick the bucket /* Polysemy: *apple*

Different kinds of linguistic knowledges

Semantics

- How words and sentences carry a meaning
- Lexical semantics: meaning relationships between words (*synonymy, antonymy, hypernymy*)
- **Compositionality**
  - Dealing with idioms: *kick the bucket /* Polysemy: *apple*

Pragmatics

- How context influences meaning
- Meaning depends on speaker intent, situation, and shared knowledge (implicit, social context)
  Can you close the window ? / It's cold in here... / Oh, great, air conditionning in winter !

Why is NLP challenging ?

Most tasks in Natural Language Processing can be viewed as resolving *ambiguity* at one of different levels:

Learning to resolve natural language ambiguities: a unified approach (Dan Roth, 1998)

Why is NLP challenging ?

Most tasks in Natural Language Processing can be viewed as resolving *ambiguity* at one of different levels:

Learning to resolve natural language ambiguities: a unified approach (Dan Roth, 1998)

- Ambiguity in written representation (speech synthesis)

Why is NLP challenging ?

Most tasks in Natural Language Processing can be viewed as resolving *ambiguity* at one of different levels:

Learning to resolve natural language ambiguities: a unified approach (Dan Roth, 1998)

- Ambiguity in written representation (speech synthesis)
- **Lexical ambiguity**, on word grammatical properties (Part-of-speech tagging) or sense (Word sense disambiguation)

Why is NLP challenging ?

Most tasks in Natural Language Processing can be viewed as resolving *ambiguity* at one of different levels:

Learning to resolve natural language ambiguities: a unified approach (Dan Roth, 1998)

- Ambiguity in written representation (speech synthesis)
- **Lexical ambiguity**, on word grammatical properties (Part-of-speech tagging) or sense (Word sense disambiguation)
- **Syntactical ambiguity** (parsing)

Why is NLP challenging ?

Most tasks in Natural Language Processing can be viewed as resolving *ambiguity* at one of different levels:

Learning to resolve natural language ambiguities: a unified approach (Dan Roth, 1998)

- Ambiguity in written representation (speech synthesis)
- **Lexical ambiguity**, on word grammatical properties (Part-of-speech tagging) or sense (Word sense disambiguation)
- **Syntactical ambiguity** (parsing)
- Ambiguity in **interpreting** a statement (sentiment classification, natural language inference)

Why is NLP challenging ?

Most tasks in Natural Language Processing can be viewed as resolving *ambiguity* at one of different levels:

Learning to resolve natural language ambiguities: a unified approach (Dan Roth, 1998)

- Ambiguity in written representation (speech synthesis)
- **Lexical ambiguity**, on word grammatical properties (Part-of-speech tagging) or sense (Word sense disambiguation)
- **Syntactical ambiguity** (parsing)
- Ambiguity in **interpreting** a statement (sentiment classification, natural language inference)

Plus, *implicit* knowledge:

- Background, **commonsense knowledge**
- Contextual knowledge

makes things difficult...

Winograd Schema Challenge

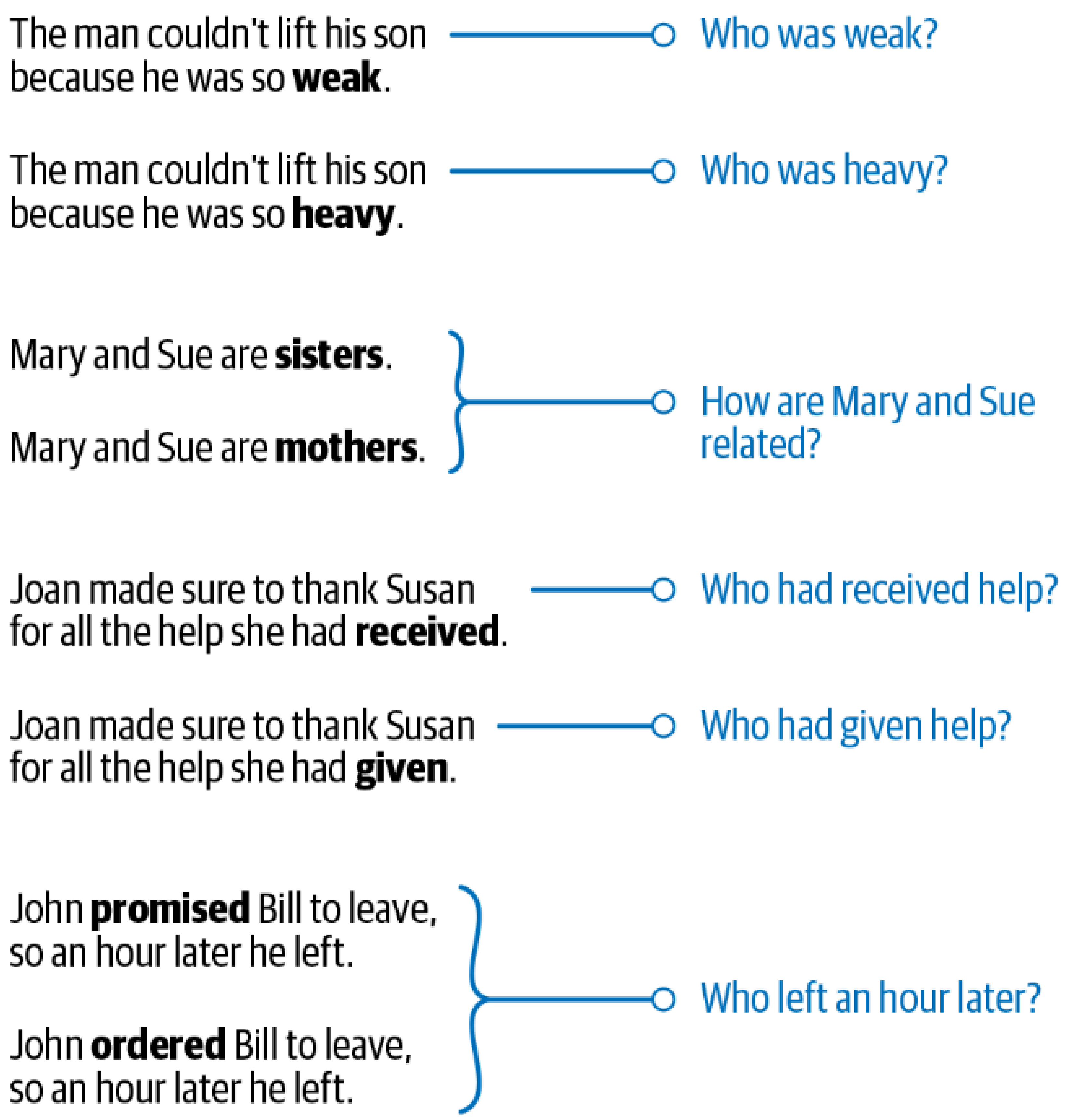
A *Winograd Schema* is a **small reading comprehension test** involving a single binary question

The Winograd Schema Challenge (Levesque et al, 2012)

Winograd Schema Challenge

A *Winograd Schema* is a **small reading comprehension test** involving a single binary question

The Winograd Schema Challenge (Levesque et al, 2012)



The man couldn't lift his son because he was so **weak**.
Who was weak?
The man couldn't lift his son because he was so **heavy**.
Who was heavy?
Mary and Sue are **sisters**.
Mary and Sue are **mothers**.
How are Mary and Sue related?
Joan made sure to thank Susan for all the help she had **received**.
Who had received help?
Joan made sure to thank Susan for all the help she had **given**.
Who had given help?
John **promised** Bill to leave, so an hour later he left.
John **ordered** Bill to leave, so an hour later he left.
Who left an hour later?

Examples of ambiguity in language from the Winograd Schema Challenge

From "Practical Language Processing, Figure 1-7, Chapter 1"

Statistical NLP

Historically, two paradigms: **symbolic** and **stochastic**

Statistical NLP

Historically, two paradigms: **symbolic** and **stochastic**

- On the symbolic side, formal systems based on logic and *grammars*

Statistical NLP

Historically, two paradigms: **symbolic** and **stochastic**

- On the symbolic side, formal systems based on logic and *grammars*
- Deep study of restricted areas: *natural language understanding* with Winograd, frameworks for *discourse modeling*

Statistical NLP

Historically, two paradigms: **symbolic** and **stochastic**

- On the symbolic side, formal systems based on logic and *grammars*
- Deep study of restricted areas: *natural language understanding* with Winograd, frameworks for *discourse modeling*
- Probabilistic models used early for tasks like *optical character recognition* regained popularity towards the end of the 80s, with models like Hidden Markov Models (HMMs) for *speech recognition*

Statistical NLP

Historically, two paradigms: **symbolic** and **stochastic**

- On the symbolic side, formal systems based on logic and *grammars*
- Deep study of restricted areas: *natural language understanding* with Winograd, frameworks for *discourse modeling*
- Probabilistic models used early for tasks like *optical character recognition* regained popularity towards the end of the 80s, with models like Hidden Markov Models (HMMs) for *speech recognition*
- Probablistic **data-driven** models then rapidly became standard

Statistical NLP

Historically, two paradigms: **symbolic** and **stochastic**

- On the symbolic side, formal systems based on logic and *grammars*
- Deep study of restricted areas: *natural language understanding* with Winograd, frameworks for *discourse modeling*
- Probabilistic models used early for tasks like *optical character recognition* regained popularity towards the end of the 80s, with models like Hidden Markov Models (HMMs) for *speech recognition*
- Probablistic **data-driven** models then rapidly became standard
- Statistical models took over rapidly from 2000, thanks to:
  - A large amount of material and resources (+computational)
  - Efficiency of statistical learning (+unsupervised approaches)
  - Community effort: shared tasks, evaluation campaigns

Statistical NLP

Historically, two paradigms: **symbolic** and **stochastic**

- On the symbolic side, formal systems based on logic and *grammars*
- Deep study of restricted areas: *natural language understanding* with Winograd, frameworks for *discourse modeling*
- Probabilistic models used early for tasks like *optical character recognition* regained popularity towards the end of the 80s, with models like Hidden Markov Models (HMMs) for *speech recognition*
- Probablistic **data-driven** models then rapidly became standard
- Statistical models took over rapidly from 2000, thanks to:
    - A large amount of material and resources (+computational)
    - Efficiency of statistical learning (+unsupervised approaches)
    - Community effort: shared tasks, evaluation campaigns
- .. and now soon represented the state-of-the-art for almost any task; now, **deep learning**

Schedule and instructions

- 12/09 - Introduction to **Text Processing and Symbolic Text Representations**
  - 19/09 - Text pre-processing, representations and visualization
- 26/09 - Introduction to **Language Modeling and Text Generation**
  - 03/10 - Introduction to language modeling
- 10/10 - **Word Embeddings**, Algorithms and Applications
  - 17/10 - Topic modeling and classification (Graded)
- 24/10 - **Sequence models**, Encoders and Decoders, Contextual Representations and Transfer Learning for NLP Tasks
  - 07/11 - Machine Translation (Graded, Mini project)
- 14/11 - **Structured Prediction** in NLP
  - 21/11 - Structured prediction

Schedule and instructions

- 28/11 - **Large Language Models** and **Societal Impact**
- 05/12 - **Sentiment Analysis**: Towards Transparent Models
- 12/12 - **Conversational Systems**

Schedule and instructions

- 28/11 - **Large Language Models** and **Societal Impact**
- 05/12 - **Sentiment Analysis**: Towards Transparent Models
- 12/12 - **Conversational Systems**

Evaluation:

- **Labs 3** (*10%*) and **4** (*20%*) will be graded
- **Exam** (*70%*, with limited personal notes)
  - Questions focused on the course: *general knowledge, methodology, good practices*
  - Small exercises *based on the labs*

Basic text processing

What is the bare minimum to process text ?

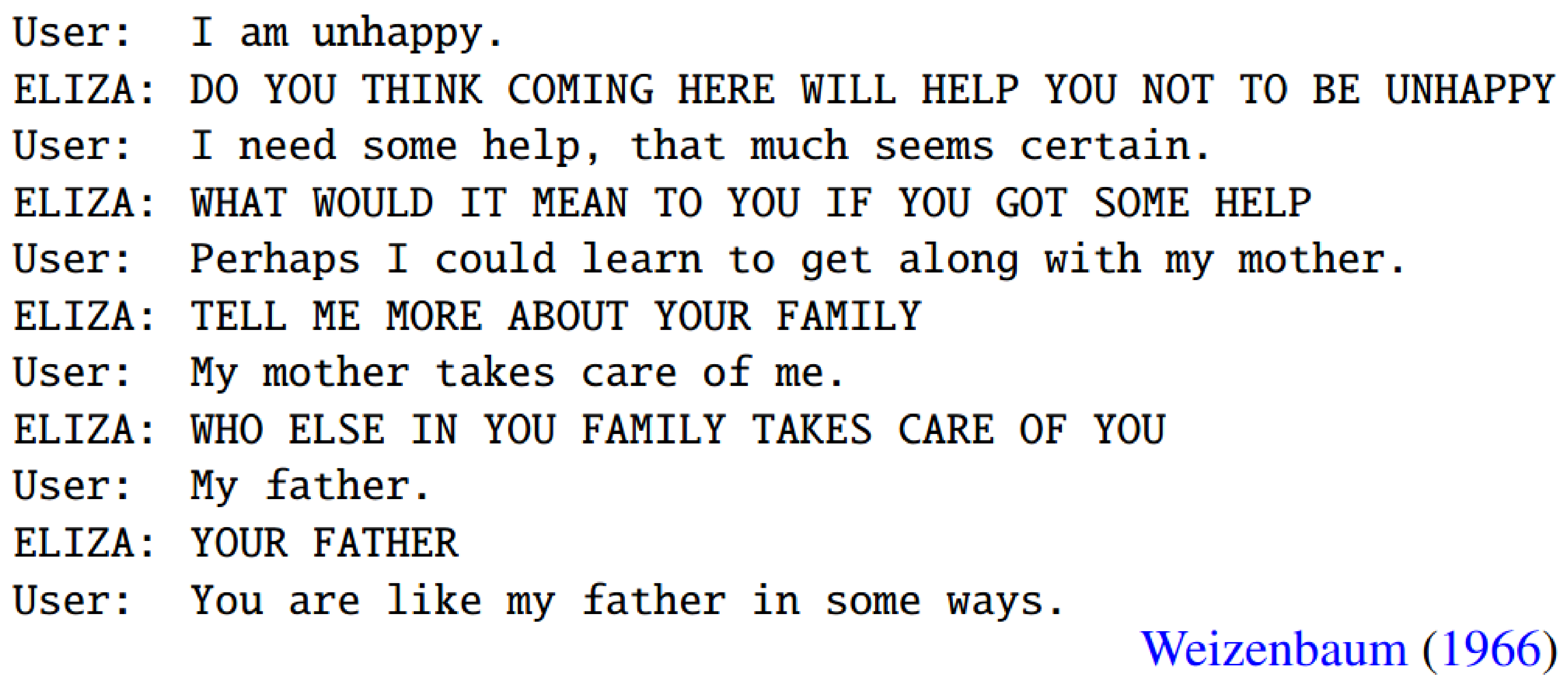
$\rightarrow$ Example of **ELIZA**

Basic text processing

What is the bare minimum to process text ?

$\rightarrow$ Example of **ELIZA**

Speech and Language Processing (Jurafsky & Manning) (Chapter II)



```
User:   I am unhappy.
ELIZA:  DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User:   I need some help, that much seems certain.
ELIZA:  WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User:   Perhaps I could learn to get along with my mother.
ELIZA:  TELL ME MORE ABOUT YOUR FAMILY
User:   My mother takes care of me.
ELIZA:  WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User:   My father.
ELIZA:  YOUR FATHER
User:   You are like my father in some ways.
```

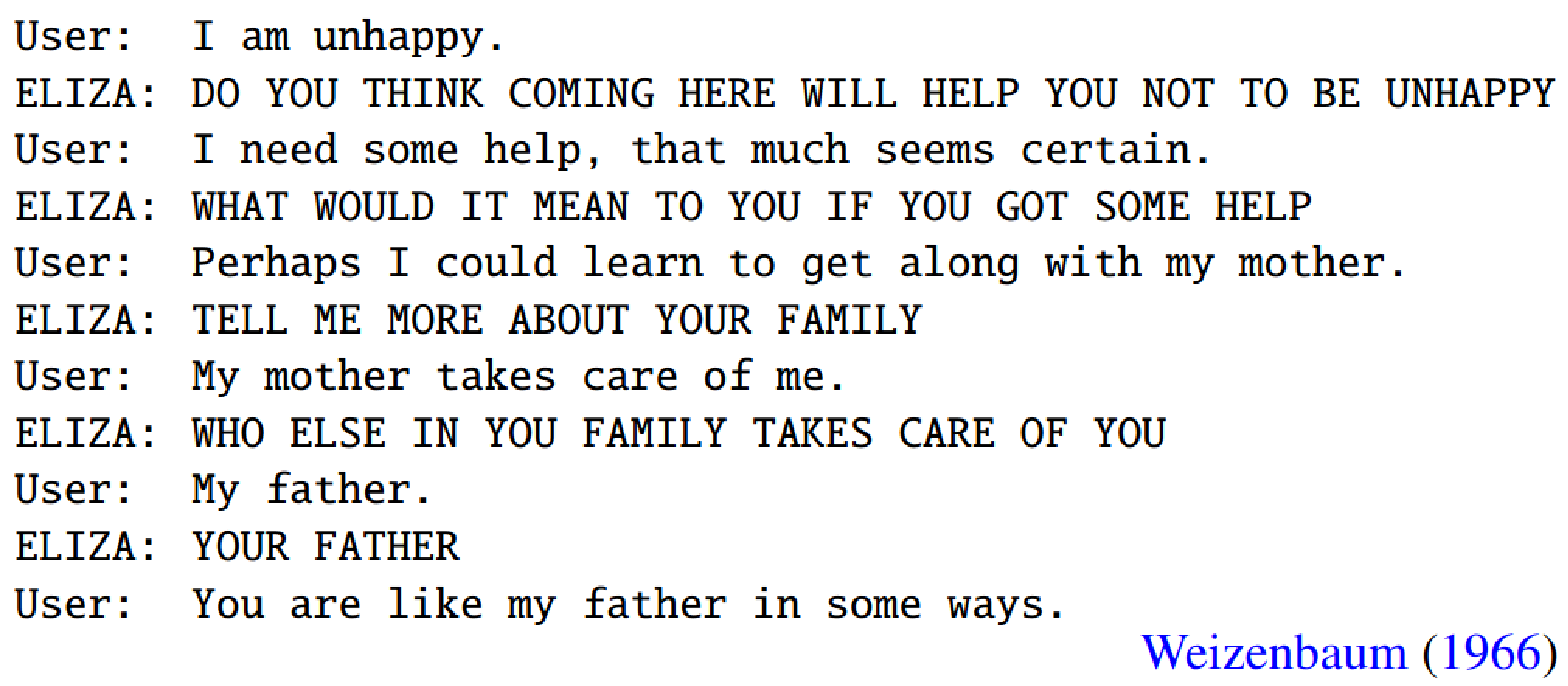
Weizenbaum (1966)

Basic text processing

What is the bare minimum to process text ?

$\rightarrow$ Example of **ELIZA**

Speech and Language Processing (Jurafsky & Manning) (Chapter II)



```
User:   I am unhappy.
ELIZA:  DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User:   I need some help, that much seems certain.
ELIZA:  WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User:   Perhaps I could learn to get along with my mother.
ELIZA:  TELL ME MORE ABOUT YOUR FAMILY
User:   My mother takes care of me.
ELIZA:  WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User:   My father.
ELIZA:  YOUR FATHER
User:   You are like my father in some ways.
```

Weizenbaum (1966)

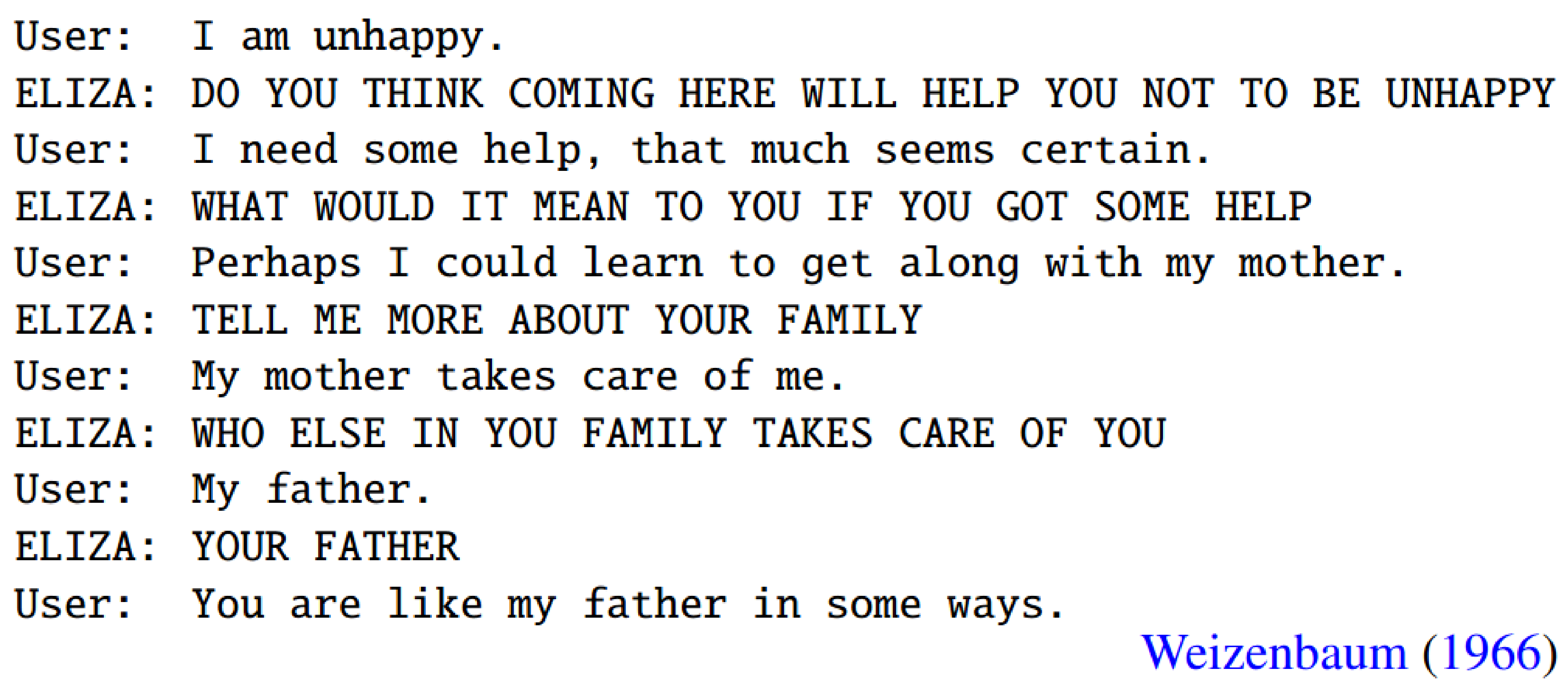
- Uses **pattern matching** to recognize phrases

Basic text processing

What is the bare minimum to process text ?

→ Example of **ELIZA**

Speech and Language Processing (Jurafsky & Manning) (Chapter II)



```
User:   I am unhappy.
ELIZA:  DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User:   I need some help, that much seems certain.
ELIZA:  WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User:   Perhaps I could learn to get along with my mother.
ELIZA:  TELL ME MORE ABOUT YOUR FAMILY
User:   My mother takes care of me.
ELIZA:  WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User:   My father.
ELIZA:  YOUR FATHER
User:   You are like my father in some ways.
```

Weizenbaum (1966)

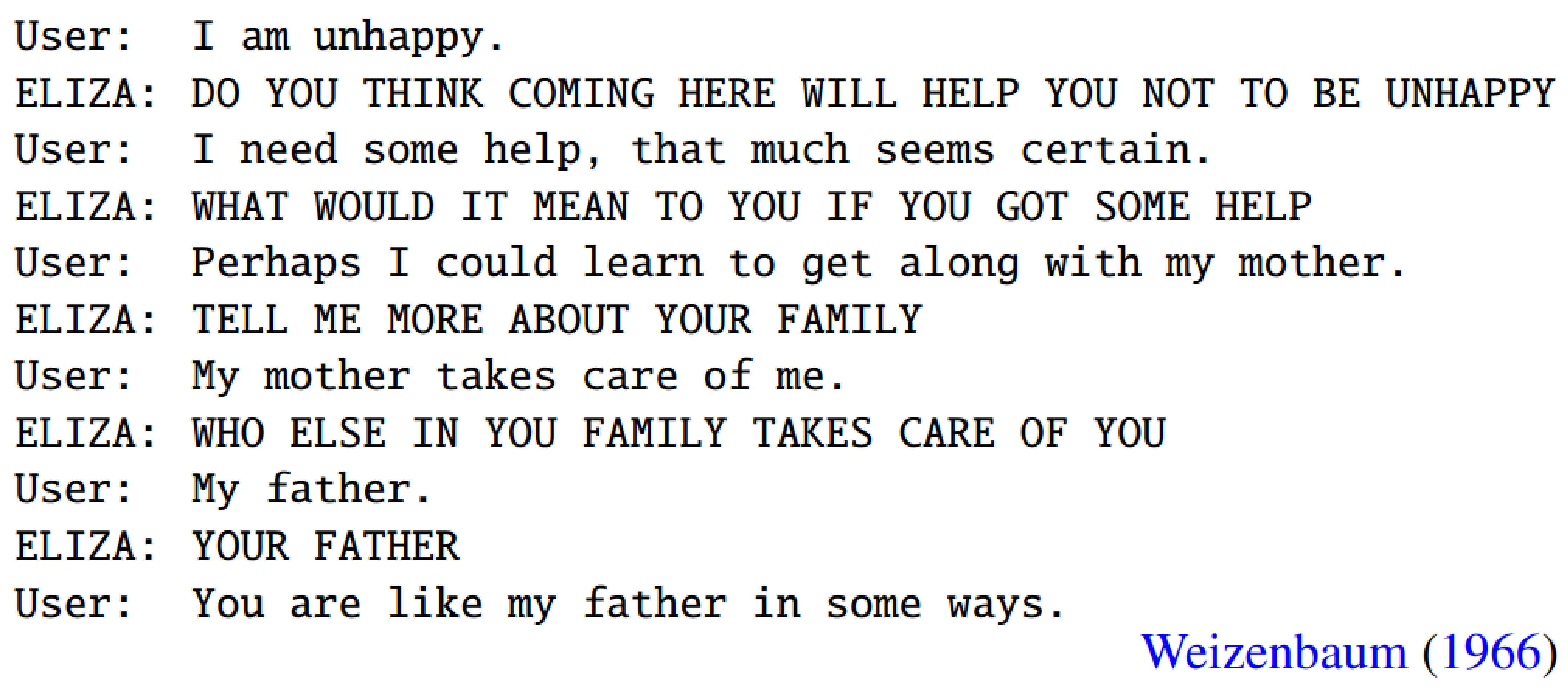
- Uses **pattern matching** to recognize phrases
- Uses a set of **rules** to translate them into suitable outputs

Basic text processing

What is the bare minimum to process text ?

$\rightarrow$ Example of **ELIZA**

Speech and Language Processing (Jurafsky & Manning) (Chapter II)



```
User:   I am unhappy.
ELIZA:  DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User:   I need some help, that much seems certain.
ELIZA:  WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User:   Perhaps I could learn to get along with my mother.
ELIZA:  TELL ME MORE ABOUT YOUR FAMILY
User:   My mother takes care of me.
ELIZA:  WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User:   My father.
ELIZA:  YOUR FATHER
User:   You are like my father in some ways.
```

Weizenbaum (1966)

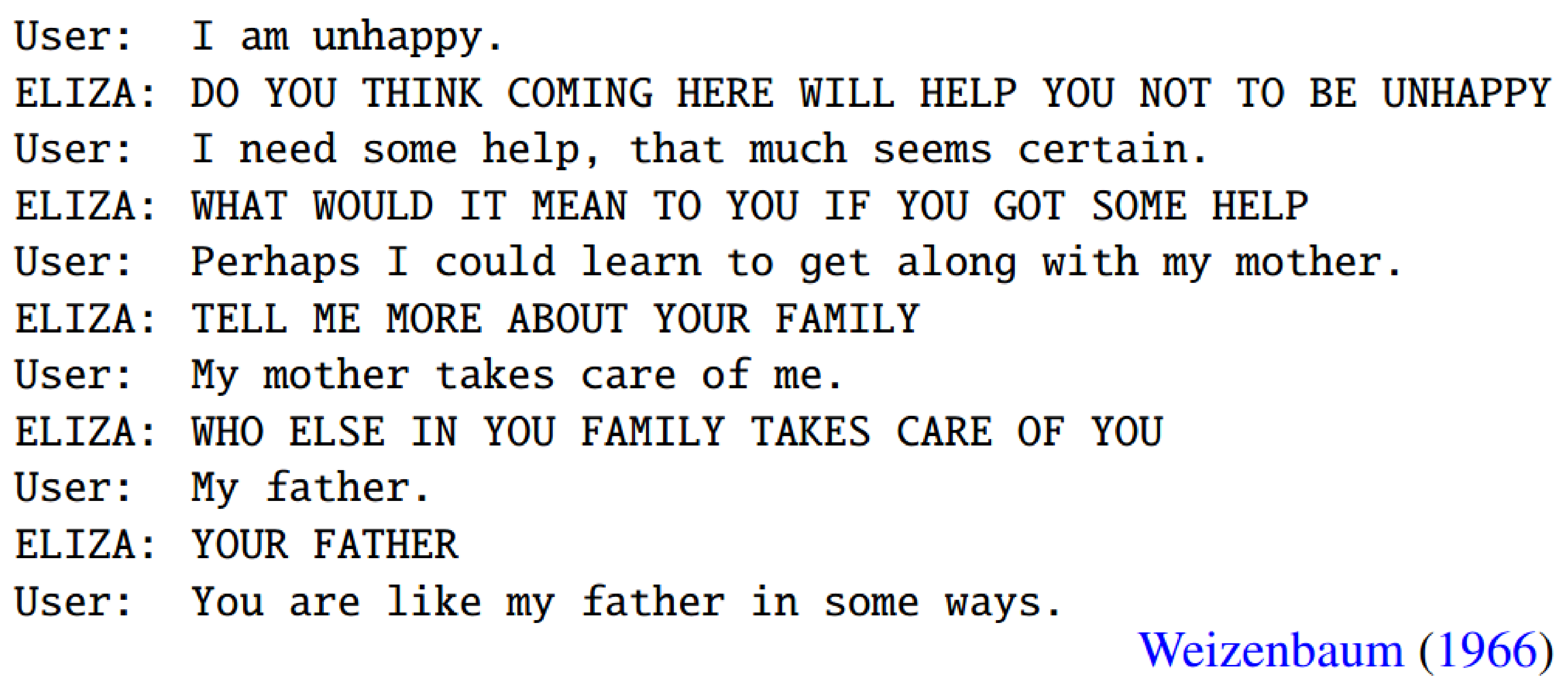
- Uses **pattern matching** to recognize phrases
- Uses a set of **rules** to translate them into suitable outputs
- Remarkably successful !

Basic text processing

What is the bare minimum to process text ?

→ Example of **ELIZA**

Speech and Language Processing (Jurafsky & Manning) (Chapter II)



```
User:   I am unhappy.
ELIZA:  DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User:   I need some help, that much seems certain.
ELIZA:  WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User:   Perhaps I could learn to get along with my mother.
ELIZA:  TELL ME MORE ABOUT YOUR FAMILY
User:   My mother takes care of me.
ELIZA:  WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User:   My father.
ELIZA:  YOUR FATHER
User:   You are like my father in some ways.
```

Weizenbaum (1966)

- Uses **pattern matching** to recognize phrases
- Uses a set of **rules** to translate them into suitable outputs
- Remarkably successful !

→ For pattern matching: **regular expressions !**

→ Used then for **text normalization** and **tokenization**

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Example: looking for '*the*':

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Example: looking for '*the*':

- Will miss some occurences:

```
the
```

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Example: looking for '*the*':

- Will miss some occurences:
- Will find undesired ones:

```
the
[tT]he
```

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Example: looking for '*the*':

- Will miss some occurences:
- Will find undesired ones:
- Will probably have very few false positives or negatives:

```
the
[tT]he
[^a-zA-Z][tT]he[^a-zA-Z]
```

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Example: looking for '*the*':

- Will miss some occurences: `the`
- Will find undesired ones: `[tT]he`
- Will probably have very few false positives or negatives: `[^a-zA-Z][tT]he[^a-zA-Z]`

$\rightarrow$ Can be used to capture and substitute text !

- Replacing '*the*' with '*The*' `s/the/The`

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Example: looking for '*the*':

- Will miss some occurences: `the`
- Will find undesired ones: `[tT]he`
- Will probably have very few false positives or negatives: `[^a-zA-Z][tT]he[^a-zA-Z]`

$\rightarrow$ Can be used to capture and substitute text !

- Replacing '*the*' with '*The*' `s/the/The`
- Capturing any string ending with '*er*': `/(.*)er/`

Regular expressions

An algebraic notation for characterizing a set of string - used practically everywhere !

$\rightarrow$ Can be used to find desired occurrences of words in a large text !

Example: looking for '*the*':

- Will miss some occurences: `the`
- Will find undesired ones: `[tT]he`
- Will probably have very few false positives or negatives: `[^a-zA-Z][tT]he[^a-zA-Z]`

$\rightarrow$ Can be used to capture and substitute text !

- Replacing '*the*' with '*The*' `s/the/The`
- Capturing any string ending with '*er*': `/(.*)er/`
- Getting a superlative: `s/the (.*)er/the (\1)est/`

Regular expressions and ELIZA

Regular expressions also allows for more complex fonctionalities.. but especially, allows for **ELIZA**:

$\rightarrow$ Early NLP system that imitated a *Rogerian psychotherapist*, by Joseph Weizenbaum (1966)

$\rightarrow$ Has a series or cascade of regular expression substitutions, matching and changing some part of the input lines:

Regular expressions and ELIZA

Regular expressions also allows for more complex fonctionalities.. but especially, allows for **ELIZA**:

$\rightarrow$ Early NLP system that imitated a *Rogerian psychotherapist*, by Joseph Weizenbaum (1966)

$\rightarrow$ Has a series or cascade of regular expression substitutions, matching and changing some part of the input lines:

- Input lines are uppercased
- Change all instances of MY to YOUR, I'M to YOU ARE, etc...
- Then, other patterns are replaced:

Regular expressions and ELIZA

Regular expressions also allows for more complex fonctionalities.. but especially, allows for **ELIZA**:

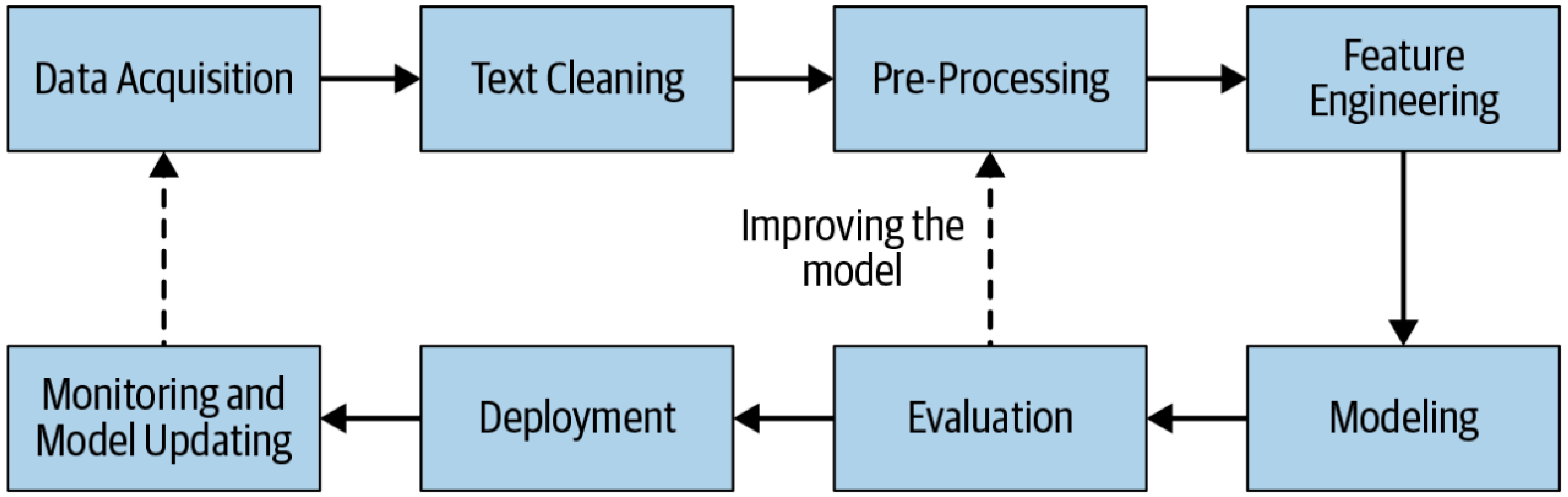
$\rightarrow$ Early NLP system that imitated a *Rogerian psychotherapist*, by Joseph Weizenbaum (1966)

$\rightarrow$ Has a series or cascade of regular expression substitutions, matching and changing some part of the input lines:

- Input lines are uppercased
- Change all instances of MY to YOUR, I'M to YOU ARE, etc...
- Then, other patterns are replaced:

```
s/.* I’M (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* I AM (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

How to build an NLP application ?



General NLP Pipeline. *From "Practical Language Processing, Figure 2-1, Chapter 2"*

Crucial steps:

- **Acquiring data** - even for rule-based system, if only for *evaluation*
- **Pre-processing**
- Feature Engineering $\rightarrow$ classical approaches *vs* modern **Text representation** through deep learning
- **Evaluation** - *intrinsic* and *extrinsic*

A few definitions

What are the *objects* we will work on ?

A few definitions

What are the *objects* we will work on ?

- We begin from a **corpus** (plural **corpora**): computer-readable collection of text, broken down into *words*

A few definitions

What are the *objects* we will work on ?

- We begin from a **corpus** (plural **corpora**): computer-readable collection of text, broken down into *words*

- **Word types** are the number of distinct words in a corpus - usually grouped in a set, the **vocabulary**

A few definitions

What are the *objects* we will work on ?

- We begin from a **corpus** (plural **corpora**): computer-readable collection of text, broken down into *words*

- **Word types** are the number of distinct words in a corpus - usually grouped in a set, the **vocabulary**
- **Word tokens** are instances of types in the running text

A few definitions

What are the *objects* we will work on ?

- We begin from a **corpus** (plural **corpora**): computer-readable collection of text, broken down into *words*

- **Word types** are the number of distinct words in a corpus - usually grouped in a set, the **vocabulary**
- **Word tokens** are instances of types in the running text

- Two types can be different **word forms** of a same **lemma:**
$\rightarrow$ *cat* and *cats* have the same lemma *cat*

A few definitions

What are the *objects* we will work on ?

- We begin from a **corpus** (plural **corpora**): computer-readable collection of text, broken down into *words*

- **Word types** are the number of distinct words in a corpus - usually grouped in a set, the **vocabulary**
- **Word tokens** are instances of types in the running text

- Two types can be different **word forms** of a same **lemma:** $\rightarrow$ *cat* and *cats* have the same lemma *cat*

$\rightarrow$ Example:

"*I showed my masterpiece to the grown−ups, and asked them whether the drawing frightened them.*"

Acquiring data: corpora and resources

Data-driven methods are based on **corpora,** which have many distinct properties:

Acquiring data: corpora and resources

Data-driven methods are based on **corpora,** which have many distinct properties:

- Language (7000+) and varieties, code switching...
- Genre (news, scientific, fiction..), specific domain (medical, law...)

- Source and authors: how was it written ? Collected ? Why ?
- Use *data statement* to avoid biases

Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science (Bender & Friedman, 2018)

Acquiring data: corpora and resources

Data-driven methods are based on **corpora,** which have many distinct properties:

- Language (7000+) and varieties, code switching...
- Genre (news, scientific, fiction..), specific domain (medical, law...)

- Source and authors: how was it written ? Collected ? Why ?
- Use *data statement* to avoid biases

Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science (Bender & Friedman, 2018)

Resources are unevenly distributed along languages ! Still, they are very diverse. For example:

- Lexical Databases like **WordNet** (but also for other languages)
- Labeled data for many tasks - will allow supervised learning
- Careful: *annotation* is a difficult and subjective process

Pre-processing: Tokenization

The first step to any task: pre-process the text, which begins by segmenting it into words. This is **tokenization !**

Pre-processing: Tokenization

The first step to any task: pre-process the text, which begins by segmenting it into words. This is **tokenization !**

- Simplest approach: *space-based* - segments along spaces
  $\rightarrow$ Does not work with some languages !

Pre-processing: Tokenization

The first step to any task: pre-process the text, which begins by segmenting it into words. This is **tokenization !**

- Simplest approach: *space-based* - segments along spaces
  $\rightarrow$ Does not work with some languages !
- What to do with punctuation ? Example:

'*But they answered: "Frighten? Why should any one be frightened by a hat?" My drawing was not a picture of a hat.*'

and many other depending on the context (hashtags, emails, etc...)

Pre-processing: Tokenization

The first step to any task: pre-process the text, which begins by segmenting it into words. This is **tokenization !**

- Simplest approach: *space-based* - segments along spaces
  $\rightarrow$ Does not work with some languages !
- What to do with punctuation ? Example:

  '*But they answered: "Frighten? Why should any one be frightened by a hat?" My drawing was not a picture of a hat.*'

  and many other depending on the context (hashtags, emails, etc...)
- Tools: we will use packages like **NLTK** and **Spacy**
  - Tokenizers are based on rules, with regular expression

Pre-processing: Tokenization

The first step to any task: pre-process the text, which begins by segmenting it into words. This is **tokenization !**

- Simplest approach: *space-based* - segments along spaces
  $\rightarrow$ Does not work with some languages !
- What to do with punctuation ? Example:

  '*But they answered: "Frighten? Why should any one be frightened by a hat?" My drawing was not a picture of a hat.*'

  and many other depending on the context (hashtags, emails, etc...)
- Tools: we will use packages like **NLTK** and **Spacy**
  - Tokenizers are based on rules, with regular expression

$\rightarrow$ After **tokenization,** the set of tokens are listed in a vocabulary $\mathcal{V}$ (*word types*)

Pre-processing: Word Normalization

We may want to choose a standard format for words:

Pre-processing: Word Normalization

We may want to choose a standard format for words:

- What to do with upper-cases ?
  $\rightarrow$ depends on the task ! Information retrieval does not need them
  - but information extraction does !

Pre-processing: Word Normalization

We may want to choose a standard format for words:

- What to do with upper-cases ?
  $\rightarrow$ depends on the task ! Information retrieval does not need them - but information extraction does !
- **Lemmatization**: represent words as their lemma
  - Done by morphological parsing
  - Necessary for some languages: *e.g*, Turkish.
  - Tools - we can also use here **NLTK**

Pre-processing: Word Normalization

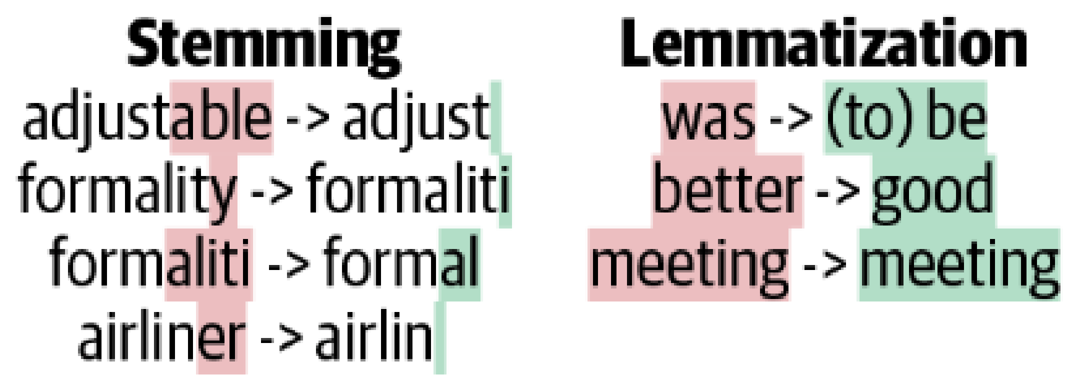
We may want to choose a standard format for words:

- What to do with upper-cases ?
  $\rightarrow$ depends on the task ! Information retrieval does not need them - but information extraction does !
- **Lemmatization**: represent words as their lemma
  - Done by morphological parsing
  - Necessary for some languages: *e.g*, Turkish.
  - Tools - we can also use here **NLTK**
- Other process: **stemming** - crudely cutting words

Pre-processing: Word Normalization

We may want to choose a standard format for words:

- What to do with upper-cases ?
  $\rightarrow$ depends on the task ! Information retrieval does not need them
  - but information extraction does !
- **Lemmatization**: represent words as their lemma
  - Done by morphological parsing
  - Necessary for some languages: *e.g*, Turkish.
  - Tools - we can also use here **NLTK**
- Other process: **stemming** - crudely cutting words



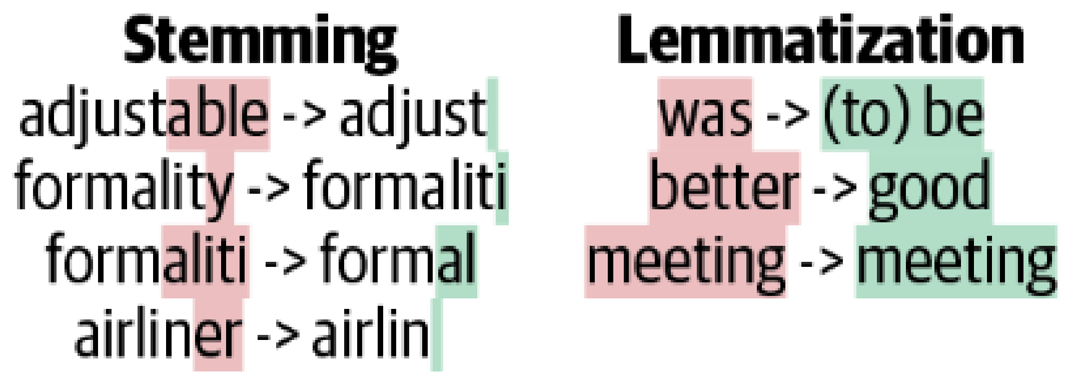
Stemming	Lemmatization
adjustable -> adjust	was -> (to) be
formality -> formaliti	better -> good
formaliti -> formal	meeting -> meeting
airliner -> airlin	

From "Practical Language Processing, Figure 2-7, Chapter 2"

Pre-processing: Word Normalization

We may want to choose a standard format for words:

- What to do with upper-cases ?
  $\rightarrow$ depends on the task ! Information retrieval does not need them - but information extraction does !
- **Lemmatization**: represent words as their lemma
  - Done by morphological parsing
  - Necessary for some languages: *e.g*, Turkish.
  - Tools - we can also use here **NLTK**
- Other process: **stemming** - crudely cutting words



Stemming	Lemmatization
adjustable -> adjust	was -> (to) be
formality -> formaliti	better -> good
formaliti -> formal	meeting -> meeting
airliner -> airlin	

From "Practical Language Processing, Figure 2-7, Chapter 2"

$\rightarrow$ Let's think ahead: what would be the advantages of the various strategies for text normalization ?

Issues with Tokenization

- **Word-level tokenization:**
  - Treats different forms of the same root as separate (e.g., “low”, “lowest”, “lowered”, etc)
  - This implies separate parameters !
  - Issues with **unknown words**

Issues with Tokenization

- **Word-level tokenization:**
  - Treats different forms of the same root as separate (e.g., “low”, “lowest”, “lowered”, etc)
  - This implies separate parameters !
  - Issues with **unknown words**
- **Character-level Tokenization:**
  - Small vocabulary, no unknown, but far longer sequences
  - How to combine character representations into representations of words meaning **must be learned**

Issues with Tokenization

- **Word-level tokenization:**
  - Treats different forms of the same root as separate (e.g., “low”, “lowest”, “lowered”, etc)
  - This implies separate parameters !
  - Issues with **unknown words**
- **Character-level Tokenization:**
  - Small vocabulary, no unknown, but far longer sequences
  - How to combine character representations into representations of words meaning **must be learned**

- A middle ground ?
  - **Byte Pair Encoding (1994)** used for **Subword Tokenization**

    Neural machine translation of rare words with subword units (Sennrich et al, 2016)
  - Now the norm for modern models

Byte Pair Encoding: Algorithm

We usually begin from *pre-tokenized* word-level tokens:

- $\mathcal{V} \leftarrow$ All characters in the training data, including an *end of word character*
- While $|\mathcal{V}| < K$:
    - Tokenize the data with the current $\mathcal{V}$
        - Taking the *longest possible prefix* each time
    - Count the frequency of adjacent token pairs in the data
    - Choose the pair $\{l, r\}$ that occurs most frequently
    - Add the pair to the vocabulary as a new token: $\mathcal{V} \leftarrow \mathcal{V} \cup \{lr\}$

The algorithm saves the **merges, in order**, and applies them in order when splitting new words

Example:

Byte Pair Encoding: Algorithm

We usually begin from *pre-tokenized* word-level tokens:

- $\mathcal{V} \leftarrow$ All characters in the training data, including an *end of word character*
- While $|\mathcal{V}| < K$:
    - Tokenize the data with the current $\mathcal{V}$
        - Taking the *longest possible prefix* each time
    - Count the frequency of adjacent token pairs in the data
    - Choose the pair $\{l, r\}$ that occurs most frequently
    - Add the pair to the vocabulary as a new token: $\mathcal{V} \leftarrow \mathcal{V} \cup \{lr\}$

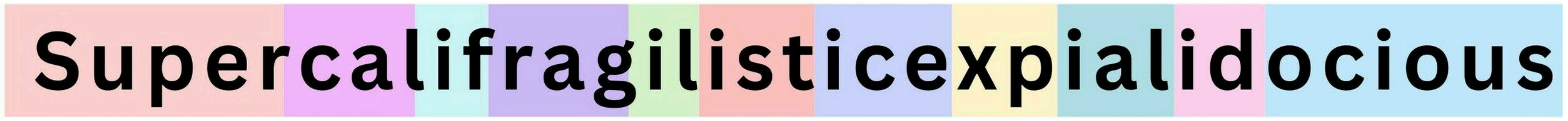
The algorithm saves the **merges, in order**, and applies them in order when splitting new words

Example:

```
l o w _ : 5
l o w e r _ : 2
n e w e s t _ : 6
w i d e s t _ : 3
```

Properties and variants

- Usually include frequent words and frequent subwords
  - Often, **morphemes** (*the smallest meaning-bearing unit of a language*)



- **WordPiece** *(Schuster et al., 2012)*:
  - Used by Google for their models
  - Merge is different (maximizes *likelihood*, not frequency)
  - Merge order is not saved: tokenization uses *the longest subword* first
- **SentencePiece** *(Kudo et al., 2018)*: applied to any algorithm
  - Replaces *whitespaces* by a special token and does not apply pre-tokenization
  - Very useful for languages without segmentation between words
- Many other variants (using *bytes, soft tokenization...*)

Sentence tokenization

What is a sentence ?

The format will heavily depend on the target task

- For many classification tasks, labels are at a higher level
  - **Labeled document**: possibly many sentences
- For core NLP tasks, it’s more complicated

Sentence tokenization

What is a sentence ?

The format will heavily depend on the target task

- For many classification tasks, labels are at a higher level
  - **Labeled document**: possibly many sentences
- For core NLP tasks, it’s more complicated
  - Ideally, it:
    - has a **complete syntactic structure**
    - is **semantically related to other sentences**

Sentence tokenization

What is a sentence ?

The format will heavily depend on the target task

- For many classification tasks, labels are at a higher level
    - **Labeled document**: possibly many sentences
- For core NLP tasks, it's more complicated
    - Ideally, it:
        - has a **complete syntactic structure**
        - is **semantically related to other sentences**
    - In practice, it is *typograhically marked*, but:
        - The full stop may be ambiguous, the uppercase too
        - Difficulty with embedded sentences (e.g, with quotes)
        - No clear markers in some languages

Tokens versus type

Type: an element of the vocabulary
Token: an instance of that type in running text

Relationship ?

Tokens versus type

Type: an element of the vocabulary
Token: an instance of that type in running text

Relationship ?

- N = Number of tokens
- $|\mathcal{V}|$ = Number of types
- **Heaps Law = Herdan's Law:**

$$|\mathcal{V}| = kN^{\beta}$$

where k and β are determined empirically

Tokens versus type

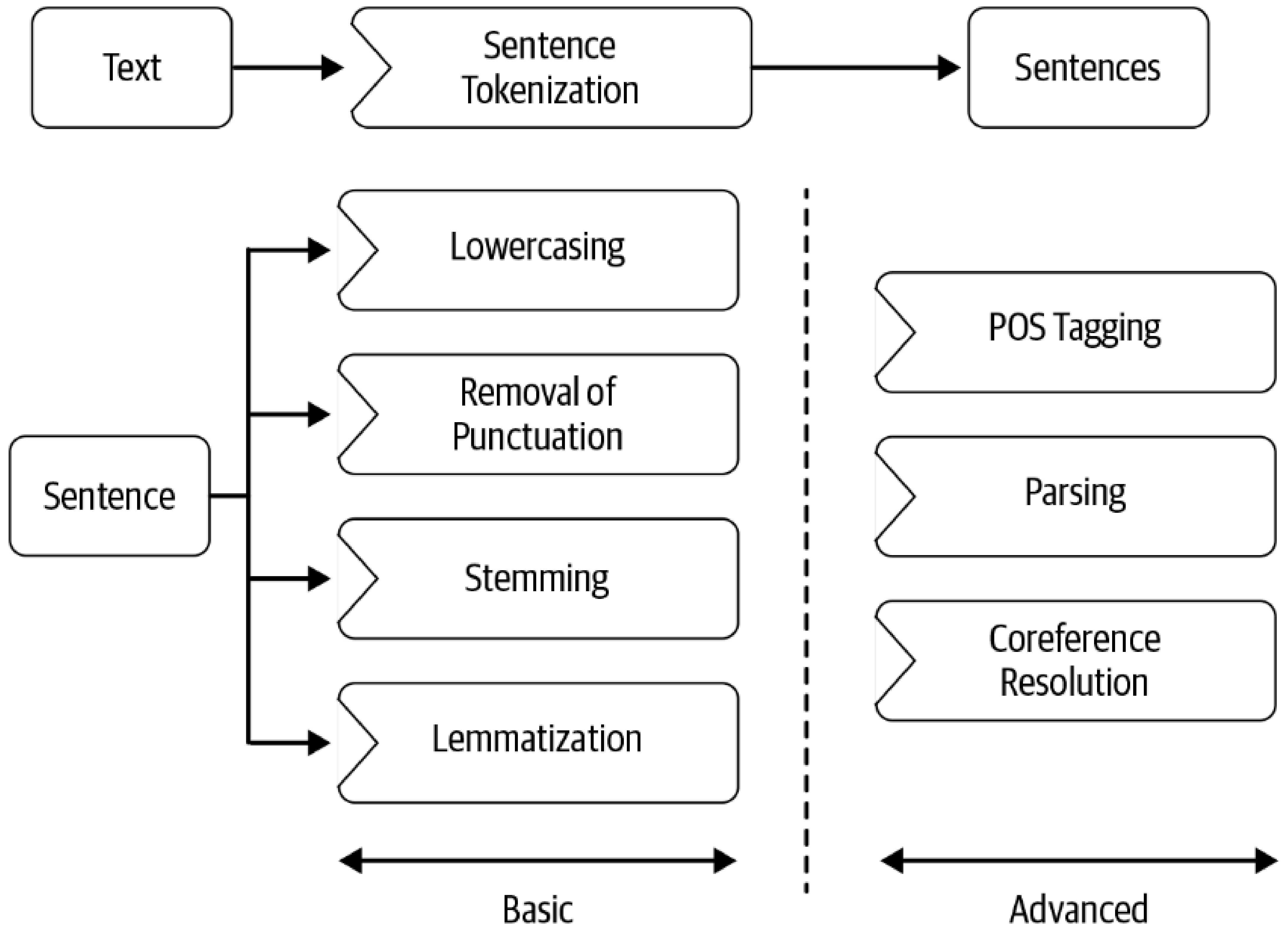
Type: an element of the vocabulary
Token: an instance of that type in running text

Relationship ?

- N = Number of tokens
- $|\mathcal{V}|$ = Number of types
- **Heaps Law = Herdan's Law:**
$$|\mathcal{V}| = kN^{\beta}$$
  where k and β are determined empirically
- For English text corpora, $10 < k < 100$ and $0.4 < \beta < 0.6$

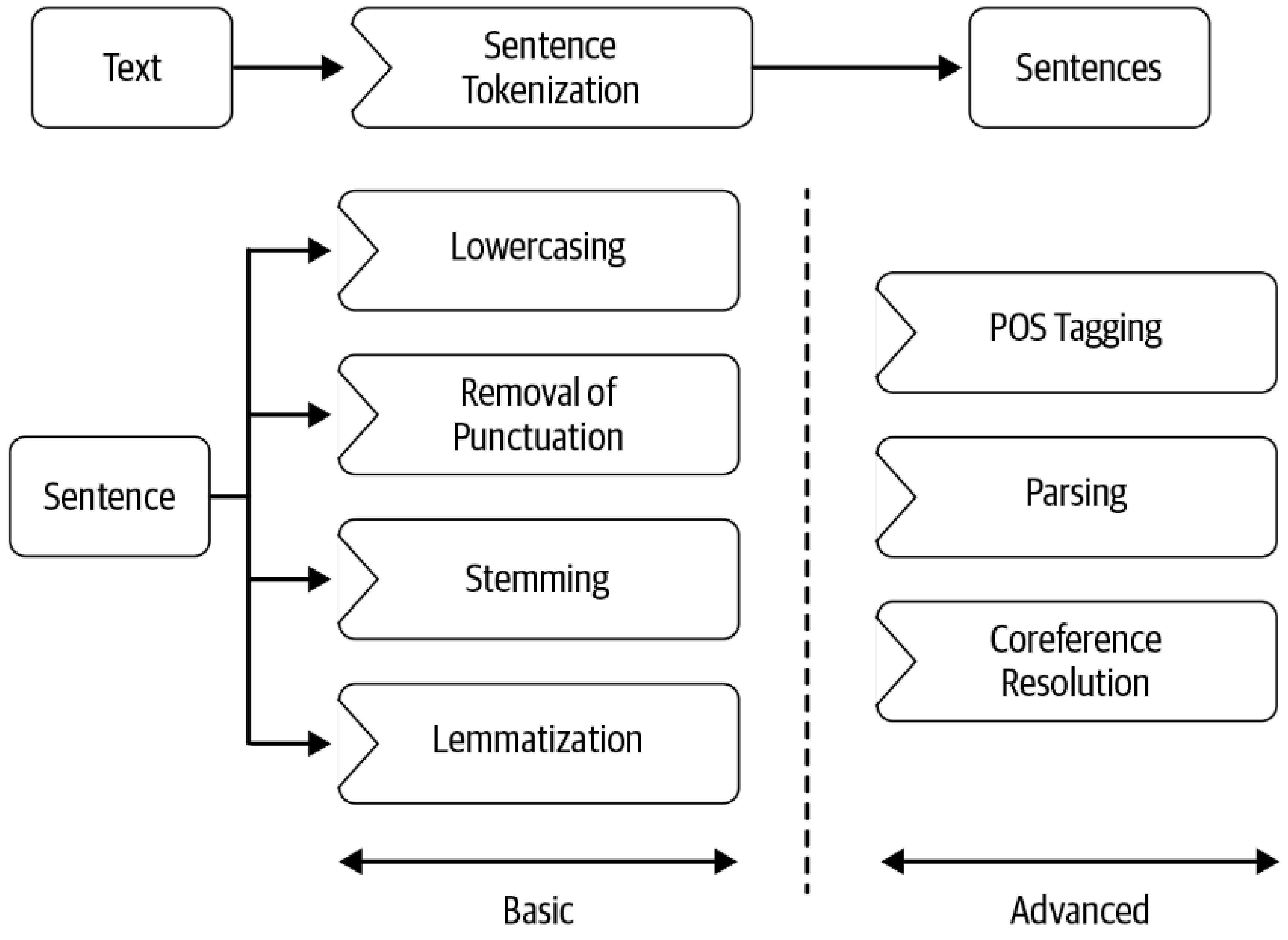
	N	$\lvert\mathcal{V}\rvert$
Switchboard	2.4 million	20 thousand
Shakespeare	884 thousand	31 thousand
Google N-gram	1 trillion	13 million

Pre-processing: summary



Summary of pre-processing steps. *From "Practical Language Processing, Figure 2-11, Chapter 2"*

Pre-processing: summary

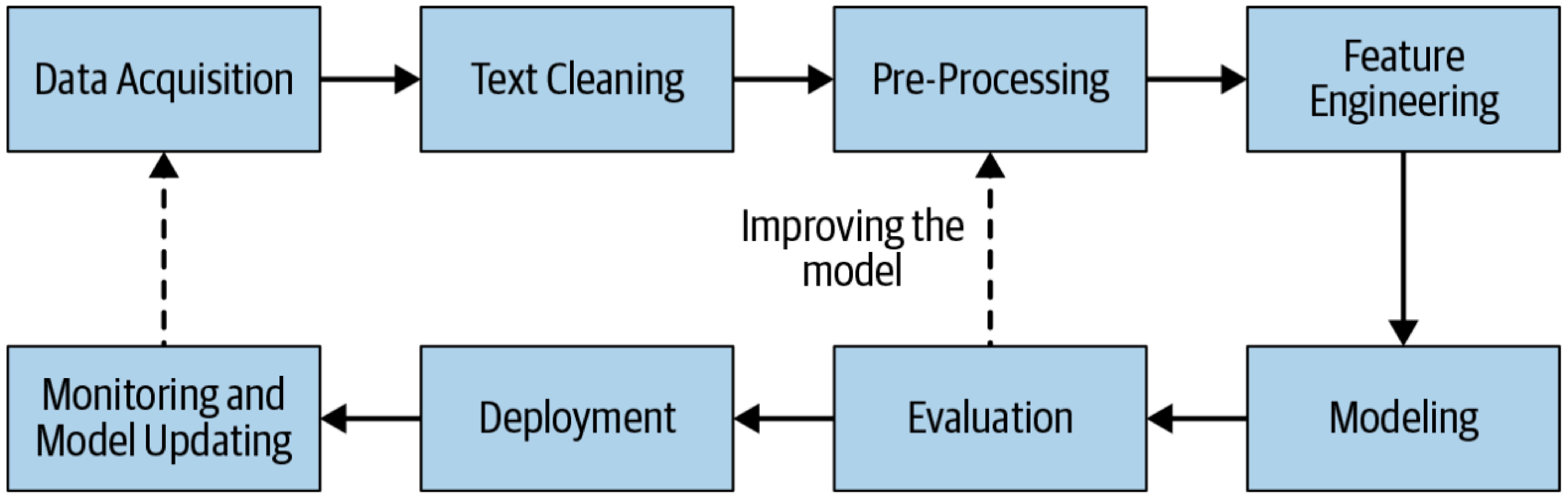


Summary of pre-processing steps. *From "Practical Language Processing, Figure 2-11, Chapter 2"*

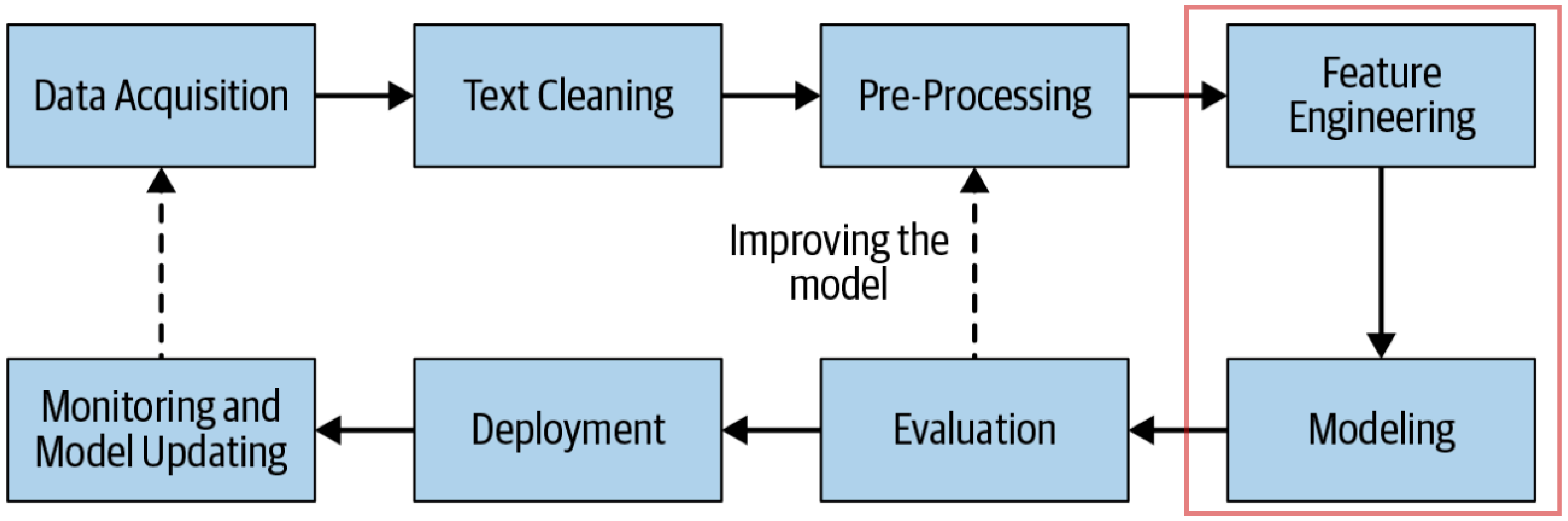
$\rightarrow$ Possible advanced steps depending on the target task !

A case study: classic methods for document representation and classification

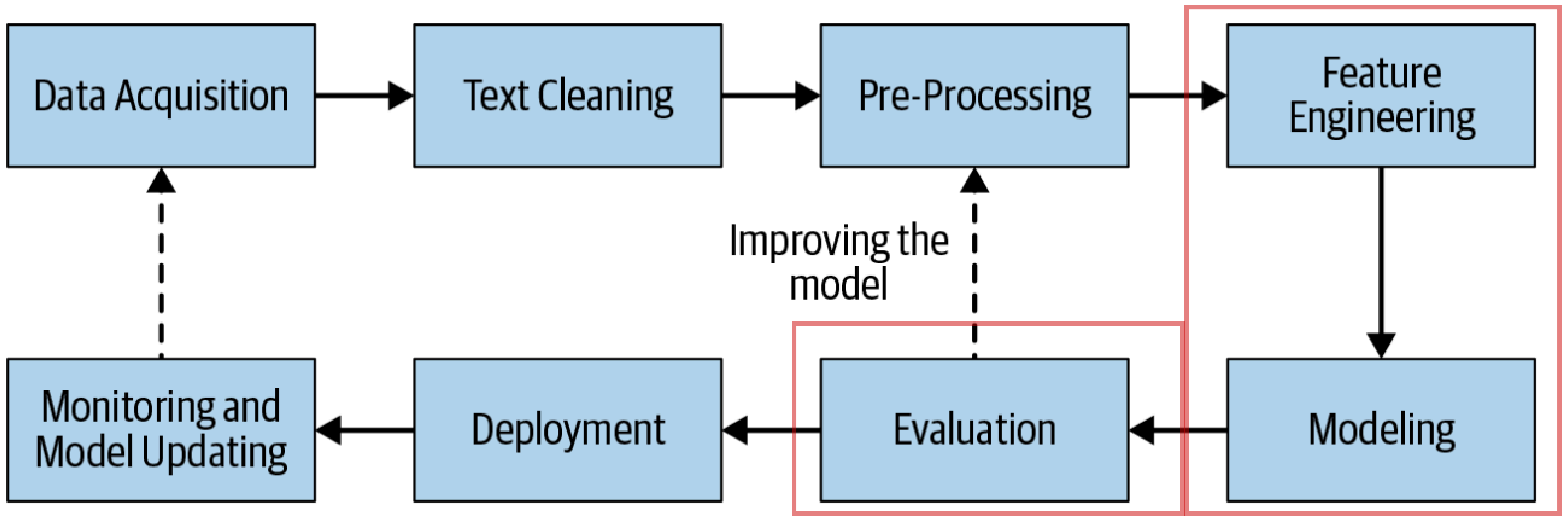
A case study: classic methods for document representation and classification



A case study: classic methods for document representation and classification



A case study: classic methods for document representation and classification



Task: Text classification

Categorizing an instance (sentence, document..) into one or more **known** classes

Task: Text classification

Categorizing an instance (sentence, document..) into one or more **known** classes

- A wide array of existing tasks, including *multi-class* and *multi-lalbel*

Task: Text classification

Categorizing an instance (sentence, document..) into one or more **known** classes

- A wide array of existing tasks, including *multi-class* and *multi-lalbel*
- Can concern sentiment, topic, ... with very different granularities
  $\rightarrow$ The label set can be structured (*hierarchical classification*)

Task: Text classification

Categorizing an instance (sentence, document..) into one or more **known** classes

- A wide array of existing tasks, including *multi-class* and *multi-lalbel*
- Can concern sentiment, topic, ... with very different granularities
  $\rightarrow$ The label set can be structured (*hierarchical classification*)
- Necessitate **labels**, hence **annotations**
  - Supervised (or semi-supervised) learning
  - Existing large scale datasets mined from user-generated content (online reviews), but it's not always the case
  - But: Human annotations are ambiguous
  - Label sets can be insufficiently defined

Task: Text classification

Categorizing an instance (sentence, document..) into one or more **known** classes

- A wide array of existing tasks, including *multi-class* and *multi-lalbel*
- Can concern sentiment, topic, ... with very different granularities
  $\rightarrow$ The label set can be structured (*hierarchical classification*)
- Necessitate **labels**, hence **annotations**
  - Supervised (or semi-supervised) learning
  - Existing large scale datasets mined from user-generated content (online reviews), but it's not always the case
  - But: Human annotations are ambiguous
  - Label sets can be insufficiently defined
- Well-defined and understood metrics, making evaluation straightforward

Features: Document as bag-of-words

The simplest way to represent a document is as a **bag-of-word** (an *unordered set of words*, keeping only their frequency).

Features: Document as bag-of-words

The simplest way to represent a document is as a **bag-of-word** (an *unordered set of words*, keeping only their frequency).

Assuming the following set of (short) documents: how to represent them ?

- *I walked down the street*
- *I walked down the avenue*
- *I ran down the street*
- *I walk down the city*
- *I walk down the avenue*

Features: Document as bag-of-words

The simplest way to represent a document is as a **bag-of-word** (an *unordered set of words*, keeping only their frequency).

Assuming the following set of (short) documents: how to represent them ?

Compute the vocabulary $\mathcal{V}$ and count !

$\Bigg\{$

- *I walked down the street*
- *I walked down the avenue*
- *I ran down the street*
- *I walk down the city*
- *I walk down the avenue*

Features: Document as bag-of-words

The simplest way to represent a document is as a **bag-of-word** (an *unordered set of words*, keeping only their frequency).

Assuming the following set of (short) documents: how to represent them ?

Compute the vocabulary $\mathcal{V}$ and count !

- *I walked down the street*
- *I walked down the avenue*
- *I ran down the street*
- *I walk down the city*
- *I walk down the avenue*

$$\rightarrow \mathcal{V} = \{\mathrm{I}, \mathrm{the}, \mathrm{down}, \mathrm{walked}, \mathrm{street}, \mathrm{avenue}, \mathrm{walk}, \mathrm{ran}, \mathrm{city}\}$$

Features: Document as bag-of-words

The simplest way to represent a document is as a **bag-of-word** (an *unordered set of words*, keeping only their frequency).

Assuming the following set of (short) documents: how to represent them ?

Compute the vocabulary $\mathcal{V}$ and count !

$$\left\{ \begin{array}{l} \end{array} \right.$$

- *I walked down the street*
- *I walked down the avenue*
- *I ran down the street*
- *I walk down the city*
- *I walk down the avenue*

$$\rightarrow \mathcal{V} = \{\mathrm{I}, \mathrm{the}, \mathrm{down}, \mathrm{walked}, \mathrm{street}, \mathrm{avenue}, \mathrm{walk}, \mathrm{ran}, \mathrm{city}\}$$

$\rightarrow \mathbf{T} =$

(Term-document matrix)

	I	the	down	walked	street	avenue	walk	ran	city
Doc_1	1	1	1	1	1	0	0	0	0
Doc_2	1	1	1	1	0	1	0	0	0
Doc_3	1	1	1	0	1	0	0	1	0
Doc_4	1	1	1	0	0	0	1	0	1
Doc_5	1	1	1	0	0	1	1	0	0

Document representation: motivation

The **Bag-of-word** is a document model counting words

- Assumes that position does not matter ... hence indifferent to syntax and semantics

Document representation: motivation

The **Bag-of-word** is a document model counting words

- Assumes that position does not matter ... hence indifferent to syntax and semantics
- Main goal: text classification (sentiment, spam ...) !

Document representation: motivation

The **Bag-of-word** is a document model counting words

- Assumes that position does not matter ... hence indifferent to syntax and semantics
- Main goal: text classification (sentiment, spam ...) !
  - We can use *rules* based on words ...
    $\rightarrow$ with *Sentiwordnet:* Each Synset is associated to a positive and negative score.

    SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining (Baccianella et al., 2010)

Document representation: motivation

The **Bag-of-word** is a document model counting words

- Assumes that position does not matter ... hence indifferent to syntax and semantics
- Main goal: text classification (sentiment, spam ...) !
  - We can use *rules* based on words ...
    $\rightarrow$ with *Sentiwordnet:* Each Synset is associated to a positive and negative score.

    SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining (Baccianella et al., 2010)
  - ...or learn a classifier model that will use *word frequencies* as features $\rightarrow$ *Naïve Bayes* is the simplest, assuming independance between words

Document representation: motivation

The **Bag-of-word** is a document model counting words

- Assumes that position does not matter ... hence indifferent to syntax and semantics
- Main goal: text classification (sentiment, spam ...) !
  - We can use *rules* based on words ...
    $\rightarrow$ with *Sentiwordnet:* Each Synset is associated to a positive and negative score.

    SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining (Baccianella et al., 2010)
  - ...or learn a classifier model that will use *word frequencies* as features $\rightarrow$ *Naïve Bayes* is the simplest, assuming independance between words
- Also useful for document clustering, information retrieval.... why ?

Model: Naïve Bayes

Multinomial naive Bayes classifier: a *generative* (why ?) linear classifier that naïvely assumes that features are independant

- Goal: for a document d return the class $\hat{c}$ with maximum a posteriori probability among classes: $\hat{c} = \mathrm{argmax}_{c \in \mathcal{C}} P(c|d)$

Model: Naïve Bayes

Multinomial naive Bayes classifier: a *generative* (why ?) linear classifier that naïvely assumes that features are independant

- Goal: for a document d return the class $\hat{c}$ with maximum a posteriori probability among classes: $\hat{c} = \mathrm{argmax}_{c \in \mathcal{C}} P(c|d)$
- Applying **Baye's rule** and the **independance assumption**, and noting $d = (w_1, ..., w_n)$ we get a (*prior* $\times$ *likelihood*) decomposition:

$$\hat{c} = \mathrm{argmax}_{c \in \mathcal{C}} \left[\mathbb{P}(c) \prod_{i=1}^{n} \mathbb{P}(w_i|c) \right]$$

  - *Derivation ?*

Model: Naïve Bayes

Multinomial naive Bayes classifier: a *generative* (why ?) linear classifier that naïvely assumes that features are independant

- Goal: for a document d return the class $\hat{c}$ with maximum a posteriori probability among classes: $\hat{c} = \mathrm{argmax}_{c \in \mathcal{C}} P(c|d)$
- Applying **Baye's rule** and the **independance assumption**, and noting $d = (w_1, ..., w_n)$ we get a (*prior* $\times$ *likelihood*) decomposition:

$$\hat{c} = \mathrm{argmax}_{c \in \mathcal{C}} \left[\mathbb{P}(c) \prod_{i=1}^{n} \mathbb{P}(w_i|c) \right]$$

  - *Derivation ?*

- Idea: we compute $\mathbb{P}(w|c)$ as the frequency of w among all documents $d \in \mathcal{D}$ of class c:

$$\mathbb{P}(w|c) = \frac{count_{\mathcal{D}}(w, c)}{\sum_{w' \in \mathcal{V}} count_{\mathcal{D}}(w', c)}$$

Model: Naïve Bayes

Multinomial naive Bayes classifier: a *generative* (why ?) linear classifier that naïvely assumes that features are independant

- Goal: for a document d return the class $\hat{c}$ with maximum a posteriori probability among classes: $\hat{c} = \mathrm{argmax}_{c \in \mathcal{C}} P(c|d)$
- Applying **Baye's rule** and the **independance assumption**, and noting $d = (w_1, ..., w_n)$ we get a (*prior* $\times$ *likelihood*) decomposition:

$$\hat{c} = \mathrm{argmax}_{c \in \mathcal{C}} \left[\mathbb{P}(c) \prod_{i=1}^{n} \mathbb{P}(w_i|c) \right]$$

  - *Derivation ?*

- Idea: we compute $\mathbb{P}(w|c)$ as the frequency of w among all documents $d \in \mathcal{D}$ of class c:

$$\mathbb{P}(w|c) = \frac{count_{\mathcal{D}}(w, c)}{\sum_{w' \in \mathcal{V}} count_{\mathcal{D}}(w', c)}$$

  - This is Maximum Likelihood Estimation (MLE): D*emonstration* ?

Classification with Naïve Bayes

- It is legitimate (and practical !) to represent a document d with its bag-of-word representation $\mathbf{d}$!
  - Practical points: use log-probabilities (why ?), use *smoothing*

Classification with Naïve Bayes

- It is legitimate (and practical !) to represent a document d with its bag-of-word representation $\mathbf{d}$!
    - Practical points: use log-probabilities (why ?), use *smoothing*

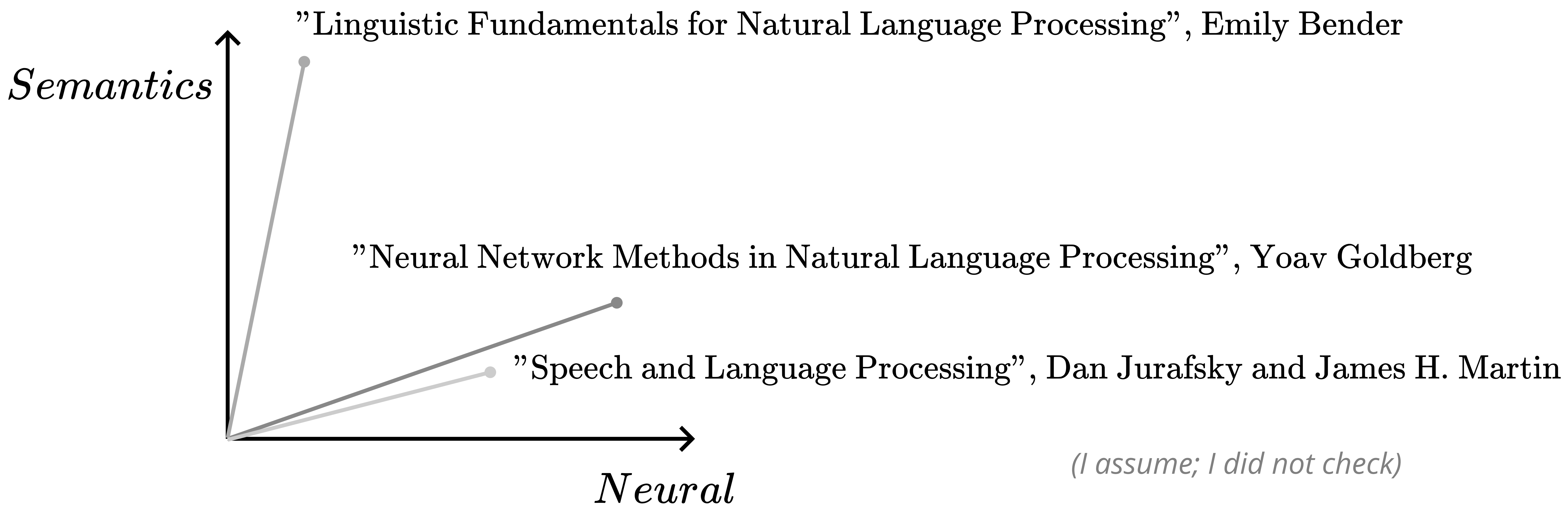
- **Training Algorithm**: Given data $\mathcal{D}$ and classes $\mathcal{C}$
    - Create the vocabulary $\mathcal{V}$ from documents $d \in \mathcal{D}$
    - From $\mathcal{D}$: For each class $c \in \mathcal{C}$: compute the log-prior $\log \mathbb{P}(c)$
        - For each word $w \in \mathcal{V}$ compute the log-likelihood $\log \mathbb{P}(w|c)$

Classification with Naïve Bayes

- It is legitimate (and practical !) to represent a document d with its bag-of-word representation $\mathbf{d}$!
    - Practical points: use log-probabilities (why ?), use *smoothing*

- **Training Algorithm**: Given data $\mathcal{D}$ and classes $\mathcal{C}$
    - Create the vocabulary $\mathcal{V}$ from documents $d \in \mathcal{D}$
    - From $\mathcal{D}$: For each class $c \in \mathcal{C}$: compute the log-prior $\log \mathbb{P}(c)$
        - For each word $w \in \mathcal{V}$ compute the log-likelihood $\log \mathbb{P}(w|c)$
- **Inference Algorithm**: Given document d to classify:
    - For each class $c \in \mathcal{C}$:$S(c) = \log \mathbb{P}(c)$
        - For each position $i \in d$:
            - If $w_i \in \mathcal{V}$: $S(c) = S(c) + \log \mathbb{P}(w_i|c)$
    - Return $\text{argmax}_{c \in \mathcal{C}} S(c)$

Document as Vectors


"Linguistic Fundamentals for Natural Language Processing", Emily Bender
Semantics
"Neural Network Methods in Natural Language Processing", Yoav Goldberg
"Speech and Language Processing", Dan Jurafsky and James H. Martin
Neural
(I assume; I did not check)

- Words are **dimension** of *documents vectors*
- You can vizualize vectors in a particular set of dimensions of your choosing
- Vectors should be *similar* for documents that are related

But what does "similar" mean here ?

Similarity between documents: cosine

The **cosine** of the angle between the document vectors is the most common similarity metric in NLP

Similarity between documents: cosine

The **cosine** of the angle between the document vectors is the most common similarity metric in NLP

- Based on the dot product - but alone, it favors long documents, with more words and higher values.

Similarity between documents: cosine

The **cosine** of the angle between the document vectors is the most common similarity metric in NLP

- Based on the dot product - but alone, it favors long documents, with more words and higher values.
- Normalizing the dot product gives us the cosine of the angle between the two vectors:

$$\mathrm{cosine}(\mathbf{d}_1, \mathbf{d}_2) = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{||\mathbf{d}_1||\ ||\mathbf{d}_2||}$$

Similarity between documents: cosine

The **cosine** of the angle between the document vectors is the most common similarity metric in NLP

- Based on the dot product - but alone, it favors long documents, with more words and higher values.
- Normalizing the dot product gives us the cosine of the angle between the two vectors:

$$\text{cosine}(\mathbf{d}_1, \mathbf{d}_2) = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{||\mathbf{d}_1||\ ||\mathbf{d}_2||}$$

- Values range in $[-1, 1]$; however, with frequencies as features, the similarity is always positive

Similarity between documents: cosine

The **cosine** of the angle between the document vectors is the most common similarity metric in NLP

- Based on the dot product - but alone, it favors long documents, with more words and higher values.
- Normalizing the dot product gives us the cosine of the angle between the two vectors:

$$\text{cosine}(\mathbf{d}_1, \mathbf{d}_2) = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{||\mathbf{d}_1||\ ||\mathbf{d}_2||}$$

- Values range in $[-1, 1]$; however, with frequencies as features, the similarity is always positive
- $\text{cosine}(\mathbf{d}_1, \mathbf{d}_2) = 0 \rightarrow$ the documents have no words in common

Similarity between documents: cosine

The **cosine** of the angle between the document vectors is the most common similarity metric in NLP

- Based on the dot product - but alone, it favors long documents, with more words and higher values.
- Normalizing the dot product gives us the cosine of the angle between the two vectors:

$$\text{cosine}(\mathbf{d}_1, \mathbf{d}_2) = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{||\mathbf{d}_1||\ ||\mathbf{d}_2||}$$

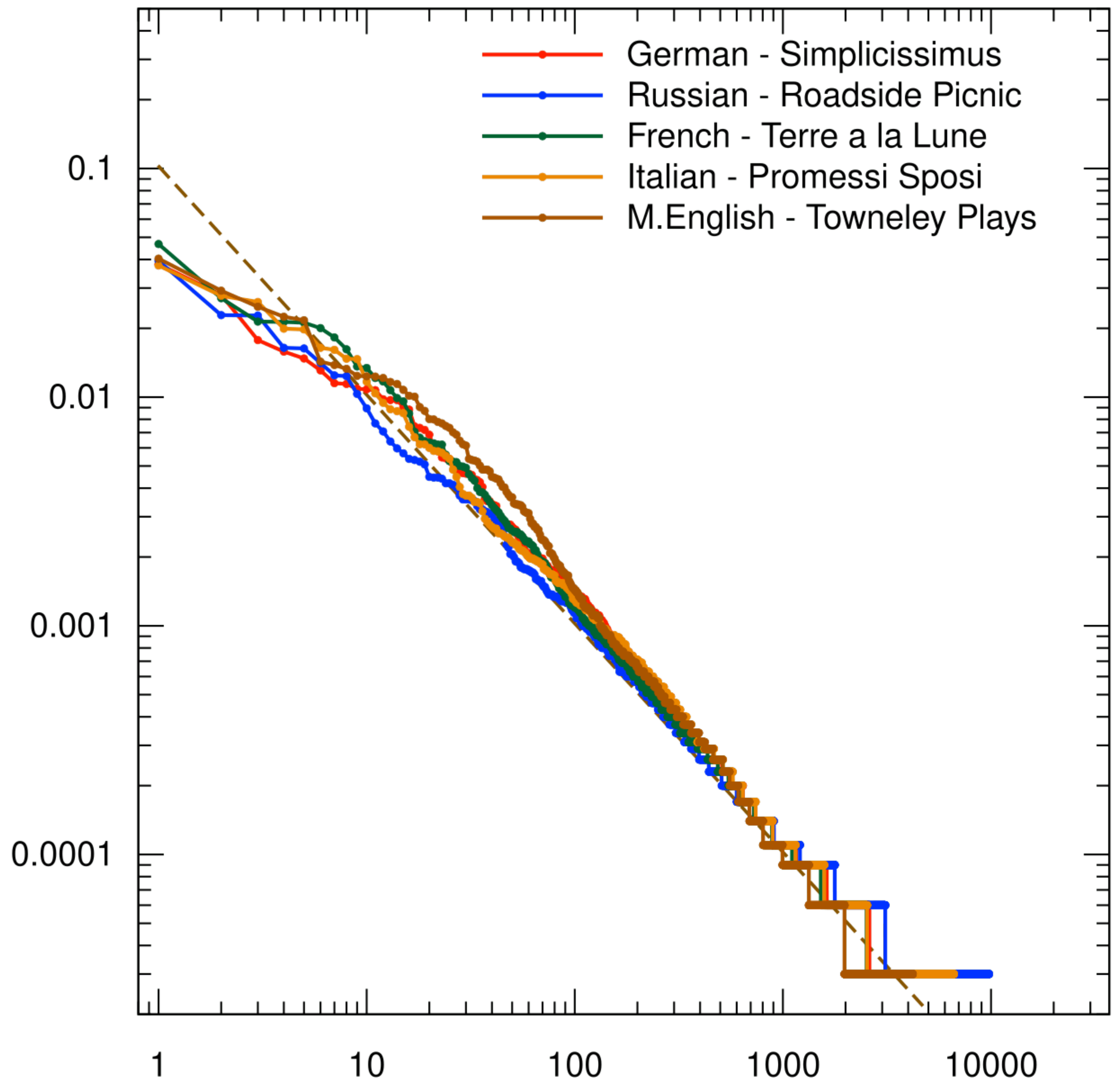
- Values range in $[-1, 1]$; however, with frequencies as features, the similarity is always positive
- $\text{cosine}(\mathbf{d}_1, \mathbf{d}_2) = 0 \rightarrow$ the documents have no words in common

Still, frequency is not the best measure of association between words:

- It is **skewed** $\rightarrow$ *Zipf*'s law
- Very frequent words are rarely the most useful for classification

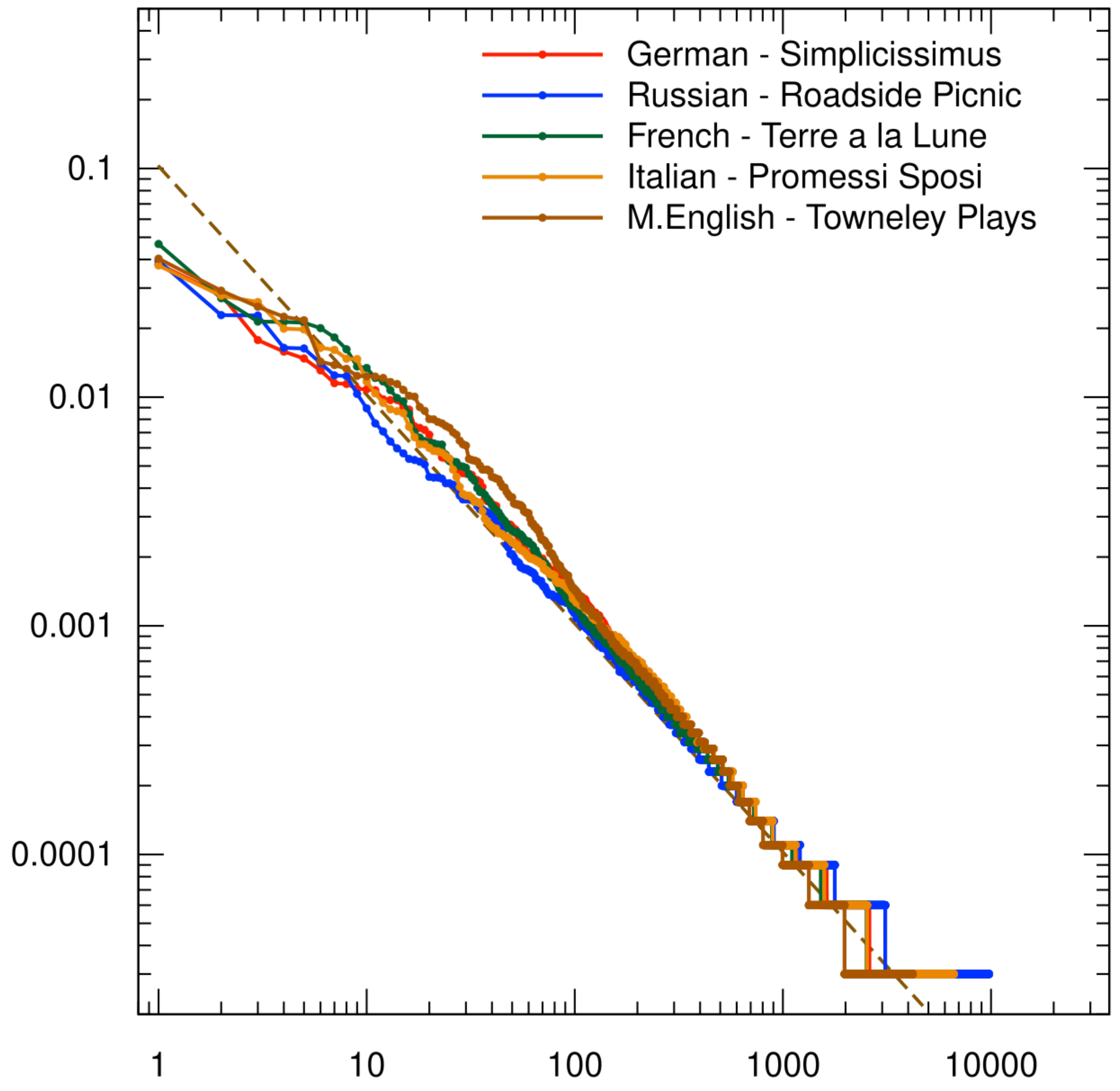
Difficulty 1: Zipf's law

Sparsity: tied to **Zipf's Law**: $frequency \propto 1/rank$



Difficulty 1: Zipf's law

Sparsity: tied to **Zipf's Law**: $frequency \propto 1/rank$



- Zipf's law seems to hold for most natural languages and many language-related phenomena
  - Examples: *meaning-frequency law, law of abbreviation*

Features: TF-IDF Representations

What would be a better way than directly removing frequent-but-not-significant words (called *stopwords*) ?

Features: TF-IDF Representations

What would be a better way than directly removing frequent-but-not-significant words (called *stopwords*) ?

- Idea 1: instead of using count $c(w, d)$ of word w in document d), define *term frequency* as $\mathrm{TF}(w, d) = \log_{10}(c(w, d) + 1)$

Features: TF-IDF Representations

What would be a better way than directly removing frequent-but-not-significant words (called *stopwords*) ?

- Idea 1: instead of using count $c(w, d)$ of word w in document d), define *term frequency* as $\mathrm{TF}(w, d) = \log_{10}(c(w, d) + 1)$
- Idea 2: give higher weight to words that occur in only a few documents, using their *inverse document frequency*. Noting $cd(w)$ the count of documents w appears in and N the total number of documents,

$$\mathrm{IDF}(w) = \log_{10}\left(\frac{N}{cd(w)}\right)$$

Features: TF-IDF Representations

What would be a better way than directly removing frequent-but-not-significant words (called *stopwords*) ?

- Idea 1: instead of using count $c(w,d)$ of word w in document d), define *term frequency* as $\mathrm{TF}(w,d) = \log_{10}(c(w,d)+1)$
- Idea 2: give higher weight to words that occur in only a few documents, using their *inverse document frequency*. Noting $cd(w)$ the count of documents w appears in and N the total number of documents,

$$\mathrm{IDF}(w) = \log_{10}\left(\frac{N}{cd(w)}\right)$$

The weight given to word w in document d is $\mathrm{TF}(w,d) \times \mathrm{IDF}(w)$:

- What happens if a word is present in every document ?

Model: Logistic regression

A *discriminative* linear classifier: learns directly $\mathbb{P}(c|d)$ through computing a **linear score** and applying a **logistic function**.

- Binary case: for a set of documents $d \in \mathcal{D}$ represented by vectors $\mathbf{d}$ learn a vector $\mathbf{w}$ and a bias b maximizing the likelihood of making a good classification into $c = 1$ or $c = 0$.

Model: Logistic regression

A *discriminative* linear classifier: learns directly $\mathbb{P}(c|d)$ through computing a **linear score** and applying a **logistic function**.

- Binary case: for a set of documents $d \in \mathcal{D}$ represented by vectors $\mathbf{d}$ learn a vector $\mathbf{w}$ and a bias b maximizing the likelihood of making a good classification into $c = 1$ or $c = 0$.
- The probability $\mathbb{P}(c = 1)$ is obtained by applying the sigmoid to the *scalar product* plus *intercept*:

$$\mathbb{P}(c = 1) = \sigma(\mathbf{w} \cdot \mathbf{d} + b)$$

Model: Logistic regression

A *discriminative* linear classifier: learns directly $\mathbb{P}(c|d)$ through computing a **linear score** and applying a **logistic function**.

- Binary case: for a set of documents $d \in \mathcal{D}$ represented by vectors $\mathbf{d}$ learn a vector $\mathbf{w}$ and a bias b maximizing the likelihood of making a good classification into $c = 1$ or $c = 0$.
- The probability $\mathbb{P}(c = 1)$ is obtained by applying the sigmoid to the *scalar product* plus *intercept*:
$$\mathbb{P}(c = 1) = \sigma(\mathbf{w} \cdot \mathbf{d} + b)$$
- We want to maximize the likelihood of our data by minimizing the **cross-entropy** between true and predicted classes $\hat{c}$ and c:
$$L(\hat{c}, c) = -\log \mathbb{P}(c|d) = -\left[c \log \hat{c} + (1-c)\log(1-\hat{c})\ \right]$$

Model: Logistic regression

A *discriminative* linear classifier: learns directly $\mathbb{P}(c|d)$ through computing a **linear score** and applying a **logistic function**.

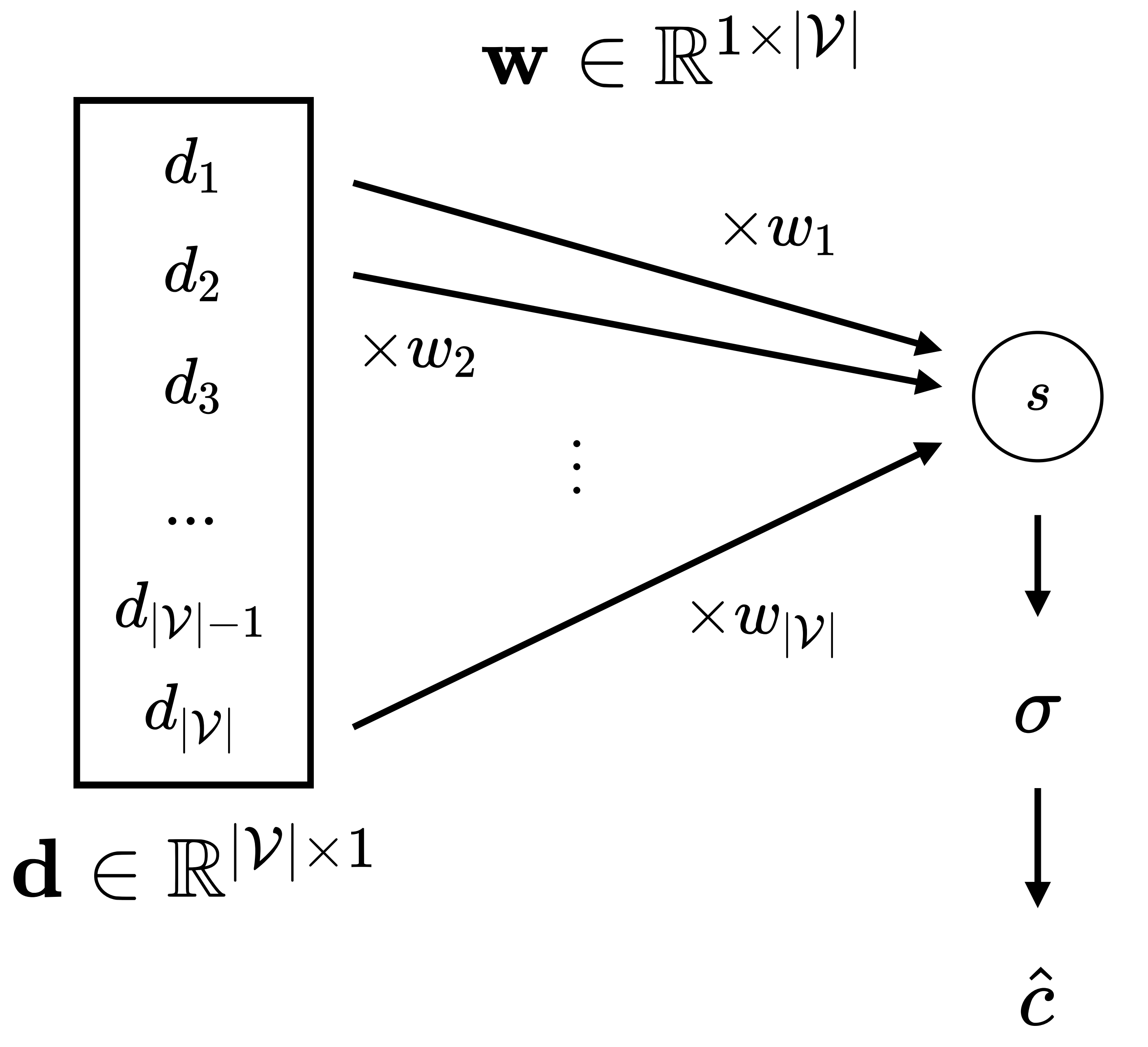
- Binary case: for a set of documents $d \in \mathcal{D}$ represented by vectors $\mathbf{d}$ learn a vector $\mathbf{w}$ and a bias b maximizing the likelihood of making a good classification into $c = 1$ or $c = 0$.
- The probability $\mathbb{P}(c = 1)$ is obtained by applying the sigmoid to the *scalar product* plus *intercept*:
$$\mathbb{P}(c = 1) = \sigma(\mathbf{w} \cdot \mathbf{d} + b)$$
- We want to maximize the likelihood of our data by minimizing the **cross-entropy** between true and predicted classes $\hat{c}$ and c:
$$L(\hat{c}, c) = -\log \mathbb{P}(c|d) = -\left[c \log \hat{c} + (1-c) \log(1-\hat{c})\ \right]$$
- Here, the training is made through **gradient descent**: we minimize that loss function by finding iteratively the direction in which the loss decreases the most and updating the weights accordingly

Classification with logistic regression

The model is easily extended to a multinomial case through using a matrix $\mathbf{W}$, a vector $\mathbf{b}$ and the *softmax* function (more on that later)

Classification with logistic regression

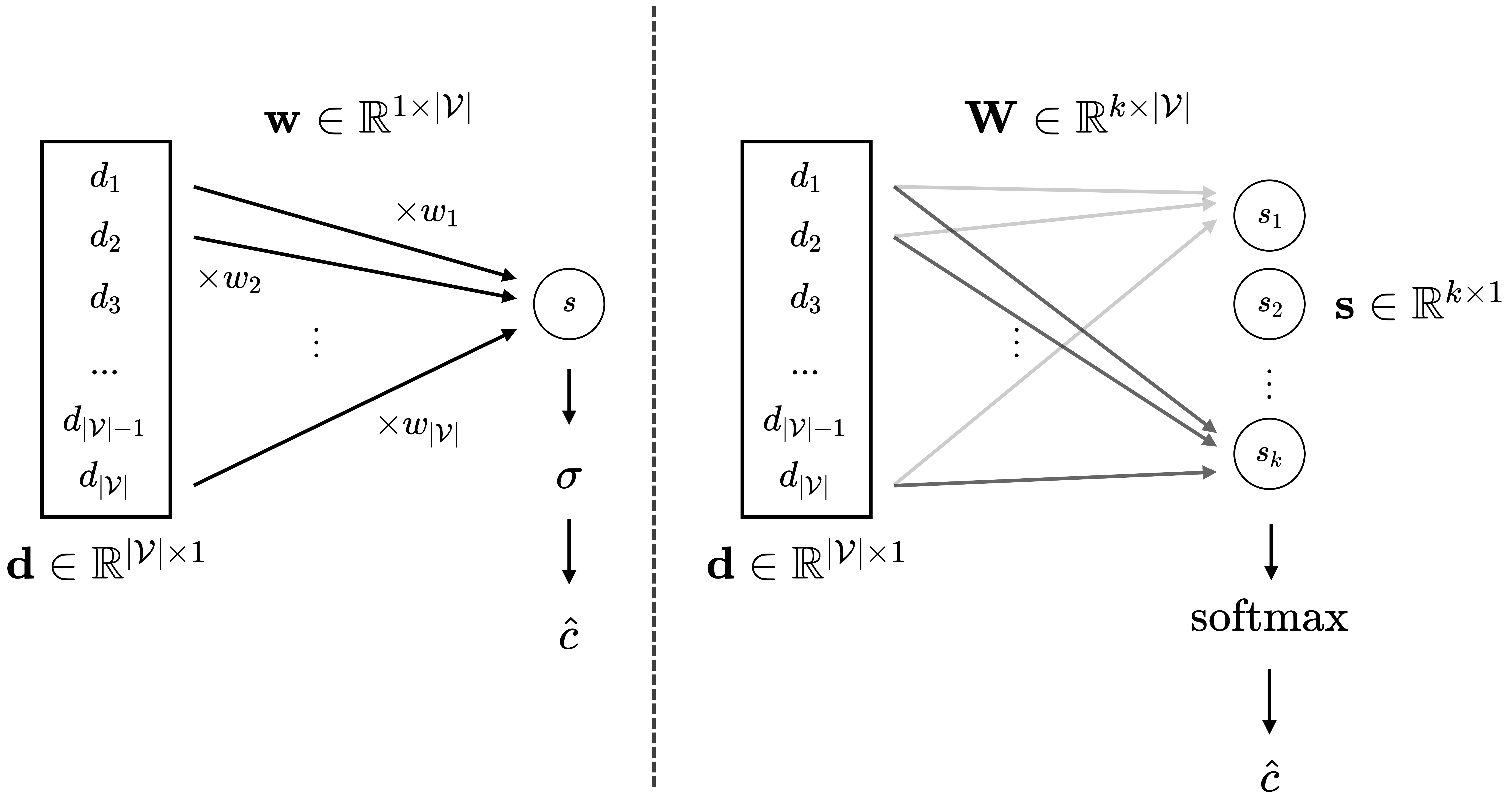
The model is easily extended to a multinomial case through using a matrix $\mathbf{W}$, a vector $\mathbf{b}$ and the *softmax* function (more on that later)



$\mathbf{w} \in \mathbb{R}^{1\times|\mathcal{V}|}$
d_1
d_2
d_3
$\dots$
$d_{|\mathcal{V}|-1}$
$d_{|\mathcal{V}|}$
$\times w_1$
$\times w_2$
$\times w_{|\mathcal{V}|}$
s
σ
$\hat{c}$
$\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|\times 1}$

Classification with logistic regression

The model is easily extended to a multinomial case through using a matrix $\mathbf{W}$, a vector $\mathbf{b}$ and the *softmax* function (more on that later)



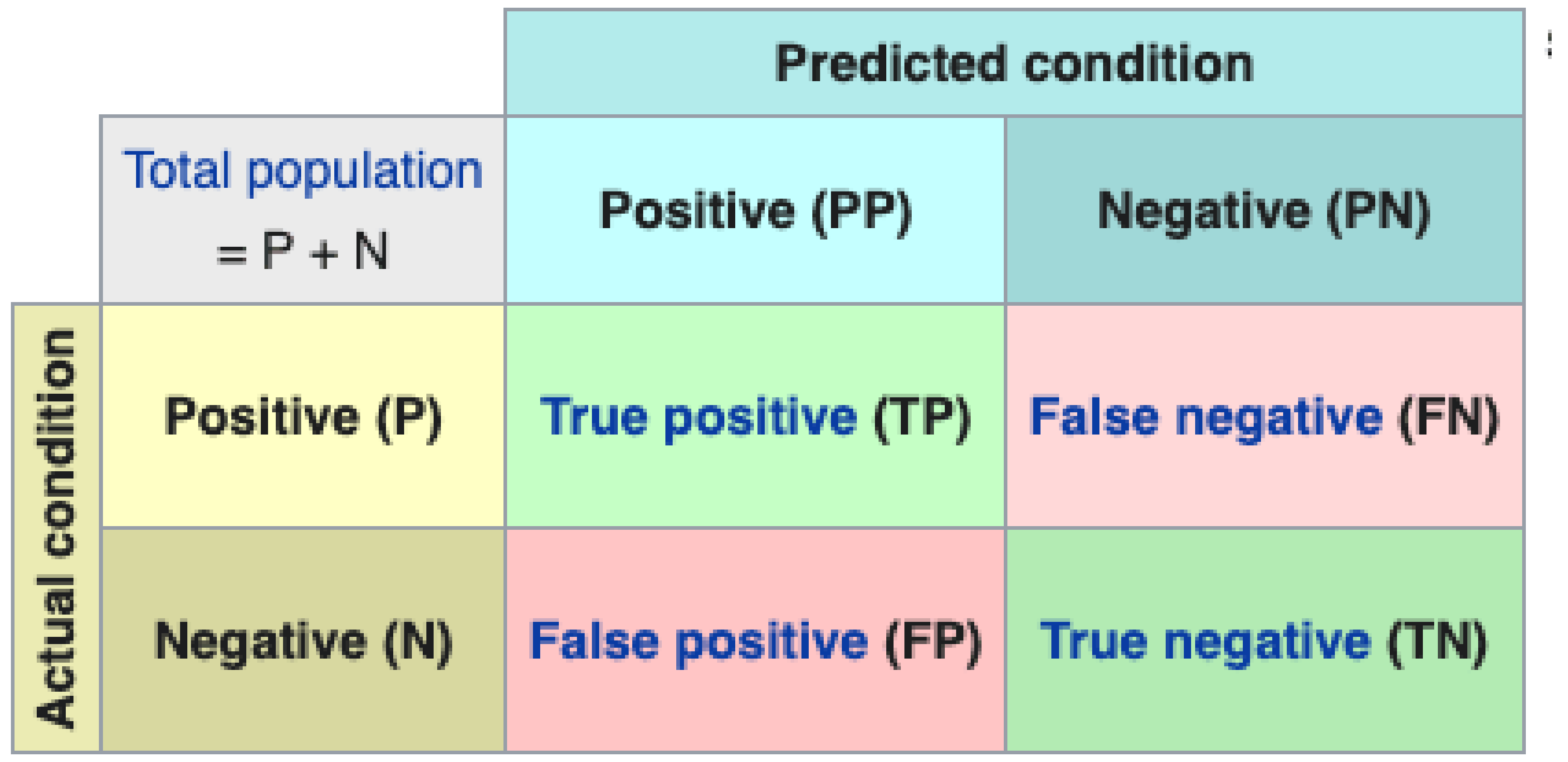
$\mathbf{w} \in \mathbb{R}^{1\times|\mathcal{V}|}$
d_1
d_2
d_3
...
$d_{|\mathcal{V}|-1}$
$d_{|\mathcal{V}|}$
$\times w_1$
$\times w_2$
$\times w_{|\mathcal{V}|}$
s
σ
$\hat{c}$
$\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|\times 1}$
$\mathbf{W} \in \mathbb{R}^{k\times|\mathcal{V}|}$
d_1
d_2
d_3
...
$d_{|\mathcal{V}|-1}$
$d_{|\mathcal{V}|}$
s_1
s_2
s_k
$\mathbf{s} \in \mathbb{R}^{k\times 1}$
softmax
$\hat{c}$
$\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|\times 1}$

Evaluation: Accuracy

- As usual: reserve held-out validation set for *hyperparameters tuning* and test set for evaluation
- Simplest measure: **Accuracy**

$$acc(\hat{\mathbf{y}},\mathbf{y}) = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}|}\delta(\hat{y}^d = y^d)$$

- For each label $c \in \mathcal{C}$, look at the *type* of
  - Errors: False positive (**FP**) and False negative (**FN**)
  - Correct predictions: True positive (**TP**) and True negative (**TN**)



		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Better metrics: F-measures and Macro

- Compute **recall** and **precision**:

$$Recall(\hat{\mathbf{y}}, \mathbf{y}, c) = \frac{TP}{TP + FN} \qquad Precision(\hat{\mathbf{y}}, \mathbf{y}, c) = \frac{TP}{TP + FP}$$

- F-measure: combines recall (r) and precision (p) using the *harmonic mean*

$$\text{F-measure}(\hat{\mathbf{y}}, \mathbf{y}, c) = \frac{2rp}{r + p}$$

- Evaluating **multi-class classification**:
  - Balance across instances: Add up **TP**, **FP**, **TN**, **FN** over classes and compute the **Micro** F-measure
  - When classes are **imbalanced**, average over classes: **Macro** F-measure:

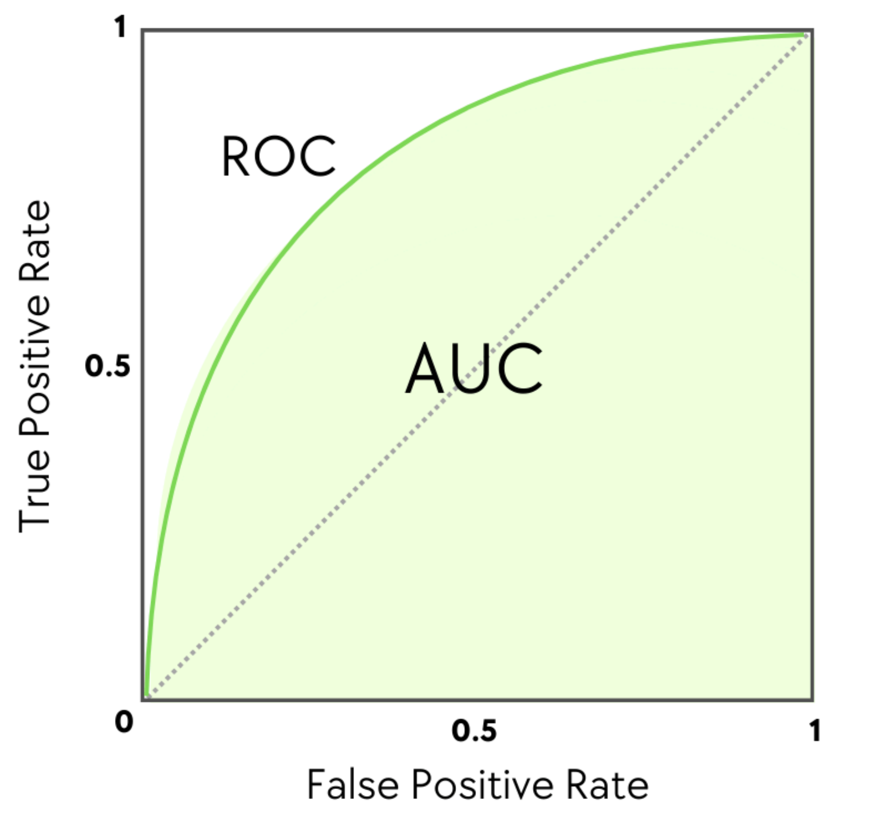
$$\text{Macro-F}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{F-measure}(\hat{\mathbf{y}}, \mathbf{y}, c)$$

AUC: Area Under the Curve

- Area Under the **ROC** (*Receiver Operating Characteristic*) Curve
  - **ROC**: $\mathrm{TPR}_s = f(\mathrm{FPR}_s)$ for different classification threshold s
    - TPR: *True Positive Ratio* $= Recall$
    - FPR: *False Positive Ratio*

$$\mathrm{FPR} = \frac{FP}{FP + TN}$$

- A metric for measuring the quality of a model **independently from the classification threshold** s
  - Usually, $s = \frac{1}{2}$
  - Can be adapted to unbalanced tasks $\rightarrow$ Anomaly detection
- The higher the AUC, the better



Back to difficulty 2: Ambiguity

To go further, NLP systems usually need to *uncover the* ***structure*** of text, which is made difficult by:

- **Lexical ambiguity**: *homography*, *polysemy*
- **Syntactic ambiguity**
- Ambiguity in **semantic scope**

Back to difficulty 2: Ambiguity

To go further, NLP systems usually need to *uncover the* ***structure*** of text, which is made difficult by:

- **Lexical ambiguity**: *homography*, *polysemy*
    - Task: Part-of-speech tagging, Word sense disambiguation
- **Syntactic ambiguity**
    - Task: Dependency parsing
- Ambiguity in **semantic scope**
    - Task: Semantic parsing

Back to difficulty 2: Ambiguity

To go further, NLP systems usually need to *uncover the* ***structure*** of text, which is made difficult by:

- **Lexical ambiguity**: *homography*, *polysemy*
  - Task: Part-of-speech tagging, Word sense disambiguation
- **Syntactic ambiguity**
  - Task: Dependency parsing
- Ambiguity in **semantic scope**
  - Task: Semantic parsing

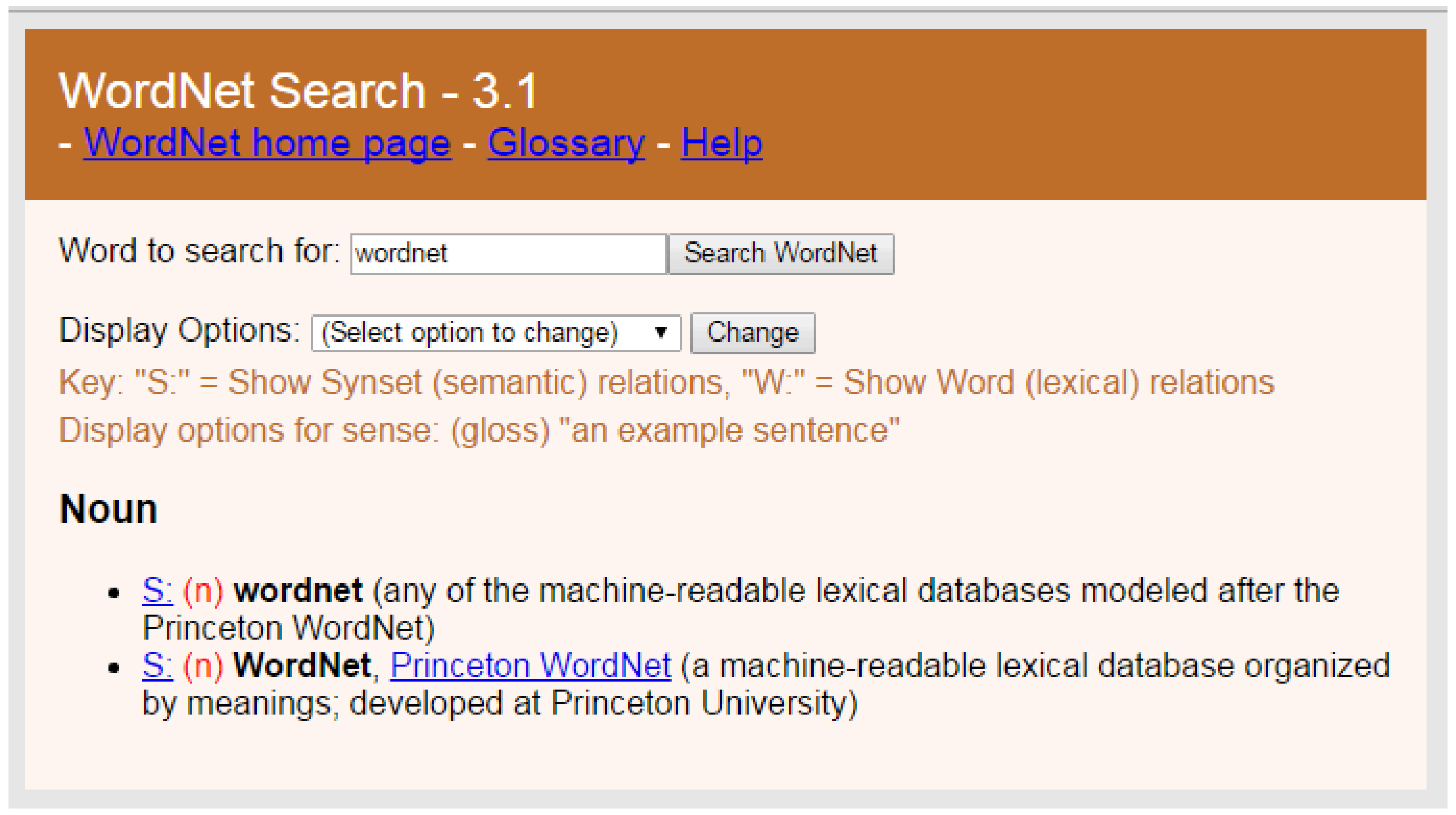
Again, things are complicated by lacking *implicit knowledge*:

- Background, commonsense knowledge
- Contextual knowledge

Features: *Implicit* representations

Beyond being counted, words can be represented by **explicit** features:

- List of **attributes** describing the object
- Natural language definition (*dictionnary*)
- Other lexical resources, including senses and associated properties, morphological features: *WordNet*



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: wordnet Search WordNet

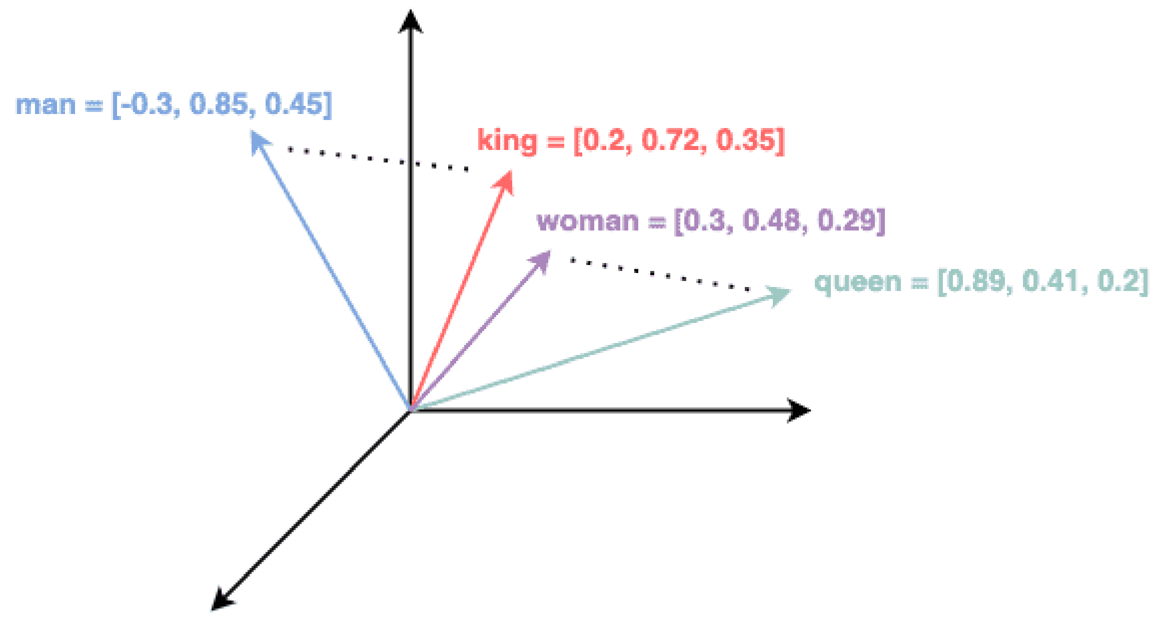
Display Options: (Select option to change) Change
Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **wordnet** (any of the machine-readable lexical databases modeled after the Princeton WordNet)
- S: (n) **WordNet**, Princeton WordNet (a machine-readable lexical database organized by meanings; developed at Princeton University)

$\rightarrow$ Move to *implicit* features: with **distributed representations**

- **Vector spaces** for words: encode **contextual** information
  - *Distributional hypothesis*: two words are similar if they have similar contexts
  - Create sparse then dense representations $\rightarrow$ Embeddings



Model: using context

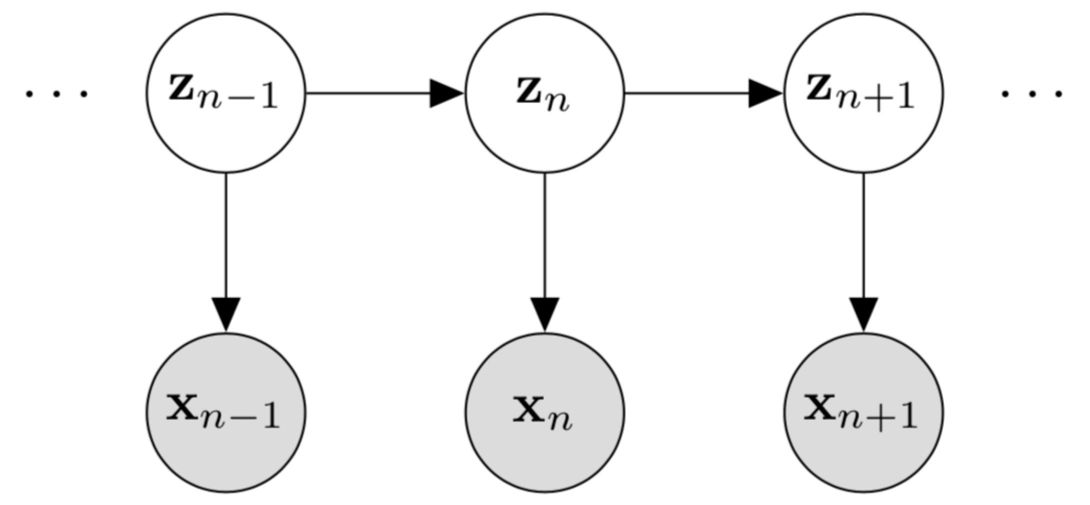
Work on modelisation of the **context** - which is, most of the time, the *surrounding sequence*:

- Use appropriate **sequence models** which will get the necessary information from the immediate context

Model: using context

Work on modelisation of the **context** - which is, most of the time, the *surrounding sequence*:

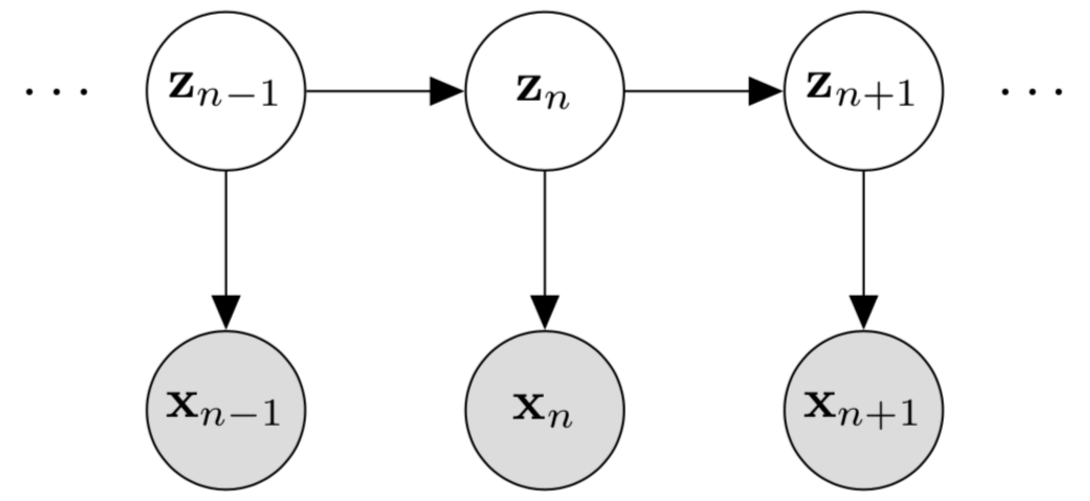
- Use appropriate **sequence models** which will get the necessary information from the immediate context
- Model **dependency within the sequence**: *Markov models*
- Generative modeling:
  - *Hidden Markov Models*



Model: using context

Work on modelisation of the **context** - which is, most of the time, the *surrounding sequence*:

- Use appropriate **sequence models** which will get the necessary information from the immediate context
- Model **dependency within the sequence**: *Markov models*
- Generative modeling:
  - *Hidden Markov Models*



- In this class: *Deep learning sequential models*
  - Architectures and objectives designed to take advantage of **large-scale unlabelled datasets**
  - Interaction with **traditional tasks**, **structures** ?
  - Integration of **exterior knowledge** ?