Basic Text Processing

Dr Muhammad Sarim

Contents

Contents

Text Normalization

- Every NLP task requires text normalization.

Text Normalization

- Every NLP task requires text normalization.
  1. Segmenting and Tokenizing words in a running text.

Text Normalization

- Every NLP task requires text normalization.
  1. Segmenting and Tokenizing words in a running text.
  2. Normalizing word formats.

Text Normalization

- Every NLP task requires text normalization.
  1. Segmenting and Tokenizing words in a running text.
  2. Normalizing word formats.
  3. Segmenting sentences in running text.

Contents

Can you count the words?

I do uh main- mainly business data processing.

Can you count the words?

I do uh main- mainly business data processing.

- Fragments, filled pauses.

Can you count the words?

I do uh main- mainly business data processing.

- Fragments, filled pauses.

Seuss's cat in the hat is different from other cats!

- Lemma: same stem, part of speech, rough word sense.
  cat and cats = same lemma.

Can you count the words?

I do uh main- mainly business data processing.

- Fragments, filled pauses.

Seuss's cat in the hat is different from other cats!

- Lemma: same stem, part of speech, rough word sense.
  cat and cats = same lemma.
- Wordform: the full inflected surface form.
  cat and cats are different wordforms.

Can you count the words?

they lay back on the San Francisco grass and looked at the stars and their

Can you count the words?

they lay back on the San Francisco grass and looked at the stars and their

- Type: an element of the vocabulary.

Can you count the words?

they lay back on the San Francisco grass and looked at the stars and their

- Type: an element of the vocabulary.
  - 13 types (or 12) (or 11?)

Can you count the words?

they lay back on the San Francisco grass and looked at the stars and their

- Type: an element of the vocabulary.
  - 13 types (or 12) (or 11?)
- Token: an instance of that type in running text.

Can you count the words?

they lay back on the San Francisco grass and looked at the stars and their

- Type: an element of the vocabulary.
  - 13 types (or 12) (or 11?)
- Token: an instance of that type in running text.
  - 15 tokens (or 14)

Can you count the words?

N = Number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

$|V| > O(\sqrt{N})$ (Church and Gale (1990))

Can you count the words?

N = Number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

$|V| > O(\sqrt{N})$ (Church and Gale (1990))

	Tokens = N	**Types = $|V|$**
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Tokenization using Unix/Linux

For a text file, output the word tokens and their frequencies.

Tokenization using Unix/Linux

For a text file, output the word tokens and their frequencies.

- change all non alphabets to new line
  `tr -sc 'A-Za-z' '\n' < mytext.txt`

Tokenization using Unix/Linux

For a text file, output the word tokens and their frequencies.

- change all non alphabets to new line
  `tr -sc 'A-Za-z' '\n' < mytext.txt`
- sort in alphabetical order
  `tr -sc 'A-Za-z' '\n' < mytext.txt | sort`

Tokenization using Unix/Linux

For a text file, output the word tokens and their frequencies.

- change all non alphabets to new line
  `tr -sc 'A-Za-z' '\n' < mytext.txt`
- sort in alphabetical order
  `tr -sc 'A-Za-z' '\n' < mytext.txt | sort`
- merge and count each type `tr -sc 'A-Za-z' '\n' < mytext.txt | sort | uniq -c`

Tokenization using Unix/Linux

For a text file, output the word tokens and their frequencies.

- change all non alphabets to new line
  tr -sc 'A-Za-z' '\n' < mytext.txt
- sort in alphabetical order
  tr -sc 'A-Za-z' '\n' < mytext.txt | sort
- merge and count each type tr -sc 'A-Za-z' '\n' < mytext.txt | sort | uniq -c
- sorting the count tr -sc 'A-Za-z' '\n' < mytext.txt | sort | uniq -c | sort -n -r

Issues in Tokenization

- Finland's capital $\rightarrow$ Finland Finlands Finland's ?

Issues in Tokenization

- Finland's capital $\rightarrow$ Finland Finlands Finland's ?
- what're, I'm, isn't $\rightarrow$ What are, I am, is not

Issues in Tokenization

- Finland's capital $\rightarrow$ Finland Finlands Finland's ?
- what're, I'm, isn't $\rightarrow$ What are, I am, is not
- Hewlett-Packard $\rightarrow$ Hewlett Packard ?

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?

Issues in Tokenization

- Finland's capital $\rightarrow$ Finland Finlands Finland's ?
- what're, I'm, isn't $\rightarrow$ What are, I am, is not
- Hewlett-Packard $\rightarrow$ Hewlett Packard ?
- state-of-the-art $\rightarrow$ state of the art ?
- Lowercase $\rightarrow$ lower-case lowercase lower case ?
- San Francisco $\rightarrow$ one token or two?

Issues in Tokenization

- Finland's capital $\rightarrow$ Finland Finlands Finland's ?
- what're, I'm, isn't $\rightarrow$ What are, I am, is not
- Hewlett-Packard $\rightarrow$ Hewlett Packard ?
- state-of-the-art $\rightarrow$ state of the art ?
- Lowercase $\rightarrow$ lower-case lowercase lower case ?
- San Francisco $\rightarrow$ one token or two?
- m.p.h., PhD. $\rightarrow$??

Issues in Tokenization: Language

- French

Issues in Tokenization: Language

- French
  - L'ensemble → one token or two?
    L ? L' ? Le ?
    Want l'ensemble to match with un ensemble

Issues in Tokenization: Language

- French
  - L'ensemble $\rightarrow$ one token or two?
    L ? L' ? Le ?
    Want l'ensemble to match with un ensemble
- German noun compounds are not segmented

Issues in Tokenization: Language

- French
  - L'ensemble → one token or two?
    L ? L' ? Le ?
    Want l'ensemble to match with un ensemble
- German noun compounds are not segmented
  - Lebensversicherungsgesellschaftsangestellter
    'life insurance company employee'
    German information retrieval needs compound splitter

Issues in Tokenization: Language

- Chinese and Japanese no spaces between words:

Issues in Tokenization: Language

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
    莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
    Sharapova now lives in US southeastern Florida

Issues in Tokenization: Language

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
    莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
    Sharapova now lives in US southeastern Florida
- Japanese has more complications, with multiple alphabets intermingled

Word Tokenization in Chinese

Also called Word Segmentation

Word Tokenization in Chinese

Also called Word Segmentation

- Chinese words are composed of characters

Word Tokenization in Chinese

Also called Word Segmentation

- Chinese words are composed of characters
  - Average word is 2.4 characters long.

Word Tokenization in Chinese

Also called Word Segmentation

- Chinese words are composed of characters
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:

Word Tokenization in Chinese

Also called Word Segmentation

- Chinese words are composed of characters
    - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
    - Maximum Matching (also known as Greedy Matching)

Maximum Matching: Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.

Maximum Matching: Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.
    1. Start a pointer at the beginning of the string

Maximum Matching: Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.
  1. Start a pointer at the beginning of the string
  2. Find the longest word in dictionary that matches the string starting at pointer

Maximum Matching: Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.
  1. Start a pointer at the beginning of the string
  2. Find the longest word in dictionary that matches the string starting at pointer
  3. Move the pointer over the word in string

Maximum Matching: Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.
  1. Start a pointer at the beginning of the string
  2. Find the longest word in dictionary that matches the string starting at pointer
  3. Move the pointer over the word in string
  4. Go to 2

Maximum Matching: Example

- Thecatinthehat $\rightarrow$ the cat in the hat

Maximum Matching: Example

- Thecatinthehat → the cat in the hat
- Thetabledownthere → the table down there ???

Maximum Matching: Example

- Thecatinthehat → the cat in the hat
- Thetabledownthere → the table down there ???

Maximum Matching: Example

- Thecatinthehat → the cat in the hat
- Thetabledownthere → the table down there ???
  theta bled own there
- Doesn't generally work in English!

Maximum Matching: Example

- Thecatinthehat → the cat in the hat
- Thetabledownthere → the table down there ???
  theta bled own there
- Doesn't generally work in English!
- But works really well in Chinese

Maximum Matching: Example

- Thecatinthehat → the cat in the hat
- Thetabledownthere → the table down there ???
  theta bled own there
- Doesn't generally work in English!
- But works really well in Chinese
- Modern probabilistic segmentation algorithms even better

Contents

Normalization

- Need to "normalize" terms

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA
- We implicitly define equivalence classes of terms

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: Asymmetric expansion:

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: Asymmetric expansion:
  - Enter: window   Search: window, windows

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: Asymmetric expansion:
  - Enter: window   Search: window, windows
  - Enter: windows   Search: Windows, windows, window

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: Asymmetric expansion:
  - Enter: window Search: window, windows
  - Enter: windows Search: Windows, windows, window
  - Enter: Windows Search: Windows

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match U.S.A. and USA
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: Asymmetric expansion:
  - Enter: window Search: window, windows
  - Enter: windows Search: Windows, windows, window
  - Enter: Windows Search: Windows
- Potentially more powerful, but less efficient

Case folding

- Applications like IR: reduce all letters to lower case

Case folding

- Applications like IR: reduce all letters to lower case
    - Since users tend to use lower case

Case folding

- Applications like IR: reduce all letters to lower case
    - Since users tend to use lower case
    - Possible exception: upper case in mid-sentence?

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - General Motors

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - General Motors
    - Fed vs. fed

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - General Motors
    - Fed vs. fed
- For sentiment analysis, MT, Information extraction

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - General Motors
    - Fed vs. fed
- For sentiment analysis, MT, Information extraction
  - Case is helpful

    US versus us is important

Lemmatization

- Reduce inflections or variant forms to base form

Lemmatization

- Reduce inflections or variant forms to base form
  - am, are, is $\rightarrow$ be

Lemmatization

- Reduce inflections or variant forms to base form
  - am, are, is $\rightarrow$ be
  - car, cars, car's, cars' $\rightarrow$ car

Lemmatization

- Reduce inflections or variant forms to base form
  - am, are, is $\rightarrow$ be
  - car, cars, car's, cars' $\rightarrow$ car
- the boy's cars are different colors $\rightarrow$ the boy car be different color

Lemmatization

- Reduce inflections or variant forms to base form
  - am, are, is $\rightarrow$ be
  - car, cars, car's, cars' $\rightarrow$ car
- the boy's cars are different colors $\rightarrow$ the boy car be different color
- Lemmatization: have to find correct dictionary headword form

Lemmatization

- Reduce inflections or variant forms to base form
  - am, are, is $\rightarrow$ be
  - car, cars, car's, cars' $\rightarrow$ car
- the boy's cars are different colors $\rightarrow$ the boy car be different color
- Lemmatization: have to find correct dictionary headword form
- Machine translation

Lemmatization

- Reduce inflections or variant forms to base form
  - am, are, is → be
  - car, cars, car's, cars' → car
- the boy's cars are different colors → the boy car be different color
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'

Contents

Morphology

- Morphemes:

Morphology

- Morphemes:
  - The small meaningful units that make up words

Morphology

- Morphemes:
  - The small meaningful units that make up words
  - Stems: The core meaning-bearing units

Morphology

- Morphemes:
  - The small meaningful units that make up words
  - Stems: The core meaning-bearing units
  - Affixes: Bits and pieces that adhere to stems
    Often with grammatical functions

Contents

Stemming

- Reduce terms to their stems in information retrieval

Stemming

- Reduce terms to their stems in information retrieval
- Stemming is crude chopping of affixes

Stemming

- Reduce terms to their stems in information retrieval
- Stemming is crude chopping of affixes
    - language dependent

Stemming

- Reduce terms to their stems in information retrieval
- Stemming is crude chopping of affixes
  - language dependent
  - e.g., automate(s), automatic, automation all reduced to automat.

Stemming

- Reduce terms to their stems in information retrieval
- Stemming is crude chopping of affixes
    - language dependent
    - e.g., automate(s), automatic, automation all reduced to automat.

Stemming

- Reduce terms to their stems in information retrieval
- Stemming is crude chopping of affixes
  - language dependent
  - e.g., automate(s), automatic, automation all reduced to automat.

for example compressed and compression are both accepted as equivalent to compress.	→	for exampl compress and compress ar both accept as equival to compress

Porters Algorithm: The most common English stemmer

- Step 1a:

sses	$\rightarrow$	ss	caresses	$\rightarrow$	caress
ies	$\rightarrow$	i	ponies	$\rightarrow$	poni
ss	$\rightarrow$	ss	caress	$\rightarrow$	caress
s	$\rightarrow$	$\emptyset$	cats	$\rightarrow$	cat

Porters Algorithm:
The most common English stemmer

- Step 1a:

sses	$\rightarrow$	ss	caresses	$\rightarrow$	caress
ies	$\rightarrow$	i	ponies	$\rightarrow$	poni
ss	$\rightarrow$	ss	caress	$\rightarrow$	caress
s	$\rightarrow$	$\emptyset$	cats	$\rightarrow$	cat

- Step 1b:

(*v*) ing	$\rightarrow$	$\emptyset$	walking	$\rightarrow$	walk
			sing	$\rightarrow$	sing
(*v*) ed	$\rightarrow$	$\emptyset$	plastered	$\rightarrow$	plaster

Porters algorithm:
The most common English stemmer

- Step 2: (long stems)

ational	$\rightarrow$	ate	relational	$\rightarrow$	relate
izer	$\rightarrow$	ize	digitizer	$\rightarrow$	digitize
ator	$\rightarrow$	ate	operator	$\rightarrow$	operate

Porters algorithm: The most common English stemmer

- Step 2: (long stems)

ational	$\rightarrow$	ate	relational	$\rightarrow$	relate
izer	$\rightarrow$	ize	digitizer	$\rightarrow$	digitize
ator	$\rightarrow$	ate	operator	$\rightarrow$	operate

- Step 3: (longer stems)

al	$\rightarrow$	$\emptyset$	revival	$\rightarrow$	reviv
able	$\rightarrow$	$\emptyset$	adjustable	$\rightarrow$	adjust
ate	$\rightarrow$	$\emptyset$	activate	$\rightarrow$	activ

Morphology in a Corpus

(*v*) ing	$\rightarrow$	$\emptyset$	walking	$\rightarrow$	walk
			sing	$\rightarrow$	sing

Morphology in a Corpus

(*v*) ing	$\rightarrow$	$\emptyset$	walking	$\rightarrow$	walk
			sing	$\rightarrow$	sing

tr -sc 'A-Za-z' '\n' < mytext.txt | grep 'ing$' | sort | uniq -c | sort -n -r

1312	King	548	being
548	being	541	nothing
541	nothing	152	something
388	king	145	coming
375	bring	130	morning
358	thing	122	having
307	ring	120	living
152	something	117	loving
145	coming	116	Being
130	morning	102	going

Morphology in a Corpus

$(*v*)\ \text{ing} \rightarrow \emptyset$ walking $\rightarrow$ walk
sing $\rightarrow$ sing

```
tr -sc 'A-Za-z' '\n' < mytext.txt | grep 'ing$' |
sort | uniq -c | sort -n -r
```

1312	King	548	being
548	being	541	nothing
541	nothing	152	something
388	king	145	coming
375	bring	130	morning
358	thing	122	having
307	ring	120	living
152	something	117	loving
145	coming	116	Being
130	morning	102	going

```
tr -sc 'A-Za-z' '\n' < mytext.txt | grep '[aeiou].*ing$' |
sort | uniq -c | sort -n -r
```

Contents

Sentence Segmentation

- !, ? are relatively unambiguous

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
    - Sentence boundary

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
    - Sentence boundary
    - Abbreviations like Inc. or Dr.
    - Numbers like .02% or 4.3

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."

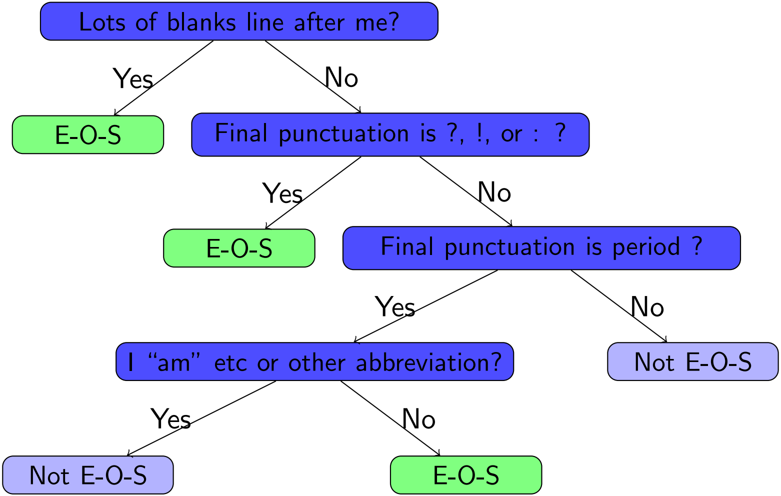
Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."
  - Decides End-Of-Sentence/Not-End-Of-Sentence

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."
  - Decides End-Of-Sentence/Not-End-Of-Sentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

A Decision Tree for E-O-S



More Sophisticated Features for DT

- Non-Numeric features

More Sophisticated Features for DT

- Non-Numeric features
  - Case of word with ".": Upper, Lower, Cap, Number

More Sophisticated Features for DT

- Non-Numeric features
  - Case of word with ".": Upper, Lower, Cap, Number
  - Case of word after ".": Upper, Lower, Cap, Number

More Sophisticated Features for DT

- Non-Numeric features
  - Case of word with ".": Upper, Lower, Cap, Number
  - Case of word after ".": Upper, Lower, Cap, Number
- Numeric features

More Sophisticated Features for DT

- Non-Numeric features
  - Case of word with ".": Upper, Lower, Cap, Number
  - Case of word after ".": Upper, Lower, Cap, Number
- Numeric features
  - Length of word with "."

More Sophisticated Features for DT

- Non-Numeric features
  - Case of word with ".": Upper, Lower, Cap, Number
  - Case of word after ".": Upper, Lower, Cap, Number
- Numeric features
  - Length of word with "."
  - Probability (word with "." occurs at E-O-S)

More Sophisticated Features for DT

- Non-Numeric features
  - Case of word with ".": Upper, Lower, Cap, Number
  - Case of word after ".": Upper, Lower, Cap, Number
- Numeric features
  - Length of word with "."
  - Probability (word with "." occurs at E-O-S)
  - Probability (word after . occurs at B-O-S)

Implementing Decision Tree

- A decision tree is just an if-then-else statement

Implementing Decision Tree

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features

Implementing Decision Tree

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand

Implementing Decision Tree

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
  - Hand-building only possible for very simple features, domains

Implementing Decision Tree

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
  - Hand-building only possible for very simple features, domains
    - For numeric features, its too hard to pick each threshold

Implementing Decision Tree

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
  - Hand-building only possible for very simple features, domains
    - For numeric features, its too hard to pick each threshold
  - Instead, structure usually learned by machine learning from a training corpus

Other Classifiers

- We can think of the questions in a decision tree

Other Classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier

Other Classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
  - Logistic regression

Other Classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
  - Logistic regression
  - Support Vector Machine

Other Classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
    - Logistic regression
    - Support Vector Machine
    - Artificial Neural Network

Other Classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
  - Logistic regression
  - Support Vector Machine
  - Artificial Neural Network
  - etc.