Agenda

- Introduction
- Natural Language Processing
- Natural Vs. Artificial (Abstract) Languages
- Computational Linguistics
- NLP Applications Area
- NLP Tasks
- Ambiguity in NLP

What is a Language?

- Language is an instrument through which we communicate. It can be symbolic, spoken, and in written form
- Language – is an exclusively human property

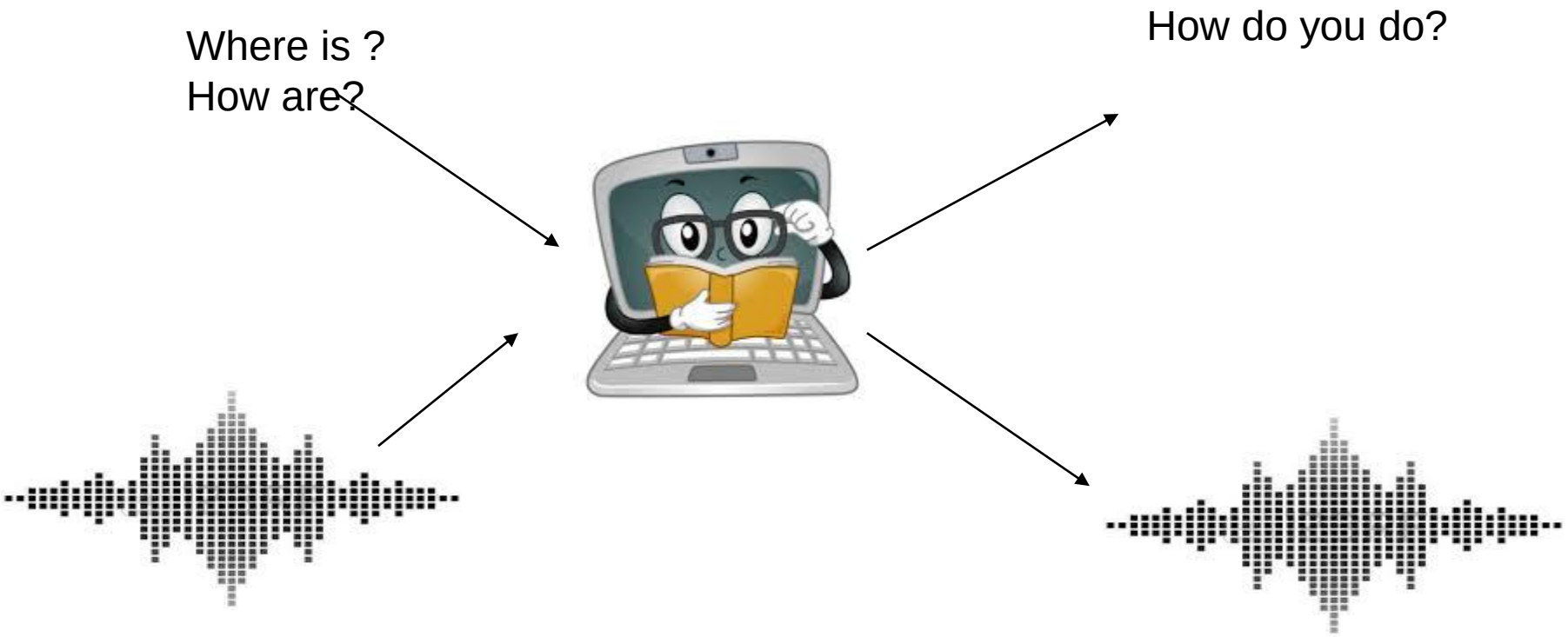
Language – Components

- Symbols set ={ 0,1}
- Words = sequence of symbols (form & meaning)
- Vocabulary = sequence of symbols or set of words
- Text = composed of sequence of words from the vocabulary
- Language = a language is constructed from sets all possible text
- A language family is a group of languages with a common origin.

Natural Language

- Natural language or ordinary language is any language that has evolved naturally in humans through use and repetition without conscious planning or premeditation
- Natural languages can take different forms, such as speech or signing. They are distinguished from constructed and formal languages such as those used to program computers

Natural Language Processing



Natural language processing (NLP) is a field of computer science concerned with the interactions between computers and human through (natural) languages interface (spoken or text forms).

Artificial Language

- Artificial languages are languages of a typically very limited size which emerge either in computer simulations between artificial agents, robot interactions or controlled psychological experiments with humans.
- It is different from formal language.
- Formal language - A formal language is a set of strings of symbols together with a set of rules that are specific to it

Natural Language vs. Artificial Language

- There are four major reasons why Natural Language (NL) is very much more difficult to process than an Artificial Language(AL)
  - NL contains a great deal of ambiguity which is controlled in AL
  - NL generally has more complex structure than is to be found in AL
  - There appears no simple universal way of representing the meaning of sentences in NL
  - Structure and meaning are necessarily interconnected in NL but not in AL

Why NLP Now?

- Four key factors enabled these developments:
  - A vast increase in computing power,
  - the availability of very large amounts of linguistic data,
  - the development of highly successful machine learning (ML) methods, and
  - a much richer understanding of the structure of human language and its deployment in social contexts.

Computational Linguistics

- The branch of linguistics in which the techniques of computer science are applied to the analysis and synthesis of language and speech
- NLP is similar to CL. In NLP computer science people use linguistics techniques to the analysis and synthesis of language and speech

Computational Linguistics

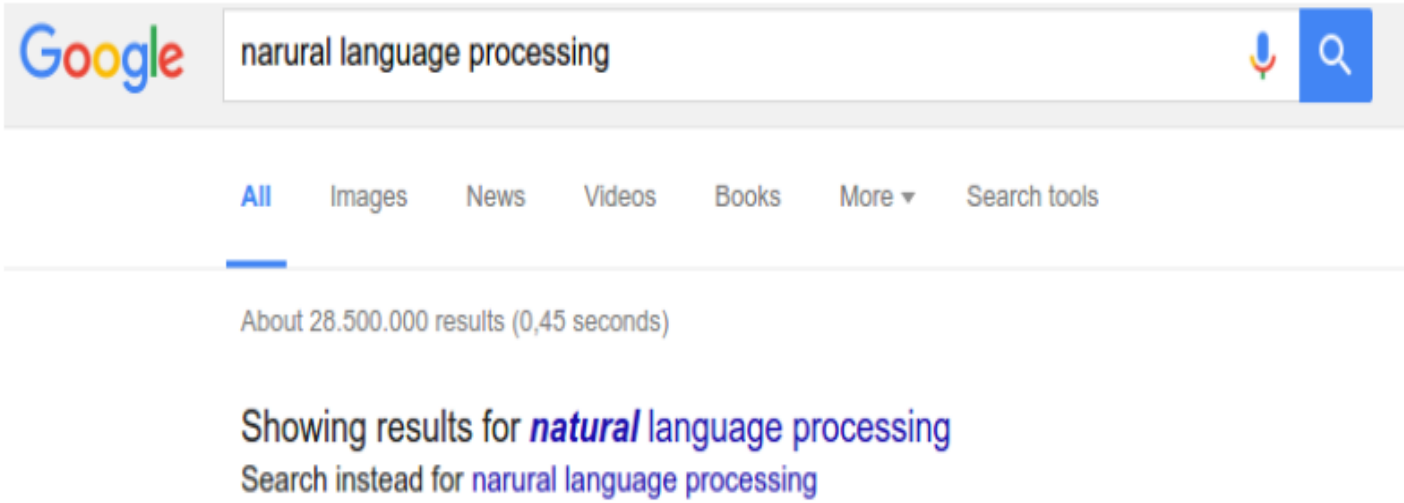
- Phonetics and Phonology— knowledge about linguistic sounds
- Morphology— knowledge of the meaningful components of words
- Syntax— knowledge of the structural relationships between words
- Semantics—knowledge of meaning
- Pragmatics— knowledge of the relationship of meaning to the goals and intentions of the speaker.
- Discourse— knowledge about linguistic units larger than a single utterance

Agenda

- Introduction
- Natural Language Processing
- Natural Vs. Artificial (Abstract) Languages
- Computational Linguistics
- NLP Applications Area
- NLP Tasks
- Ambiguity in NLP

NLP Applications

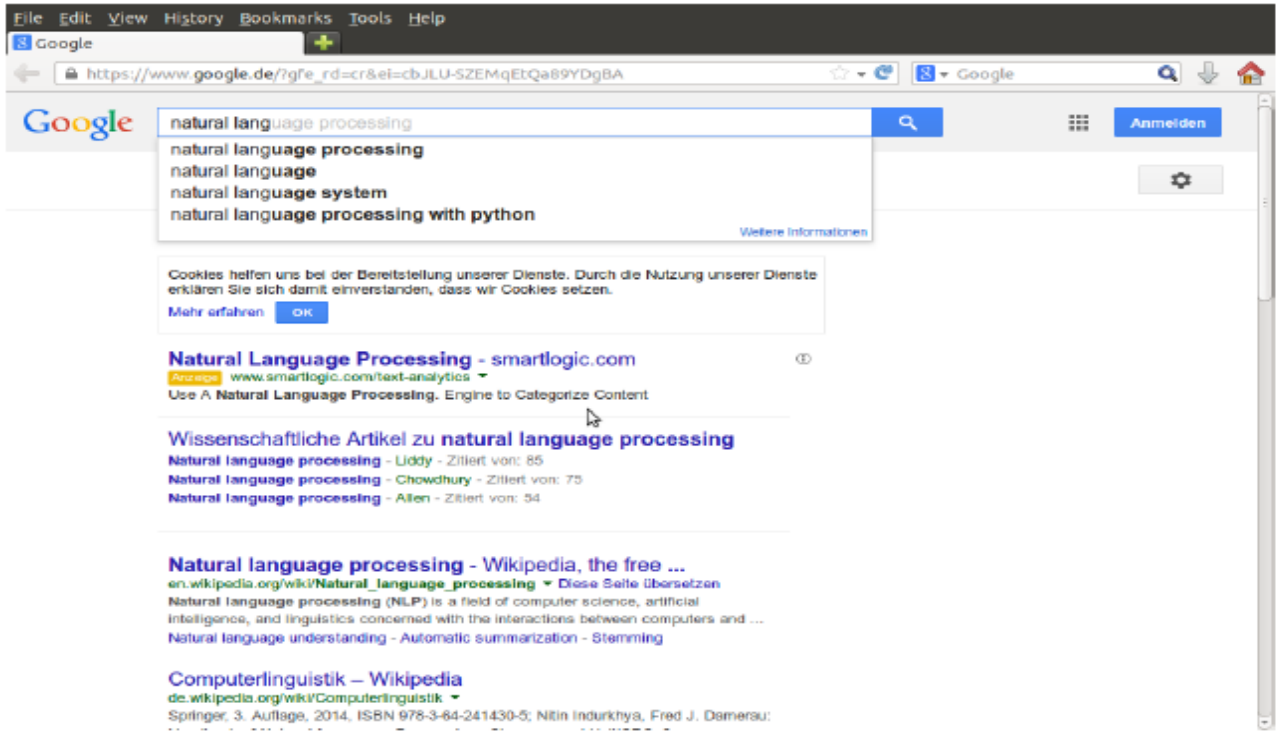
- Spelling & Grammar Checking & Corrections



Identifying errors and suggesting alternate

NLP Applications

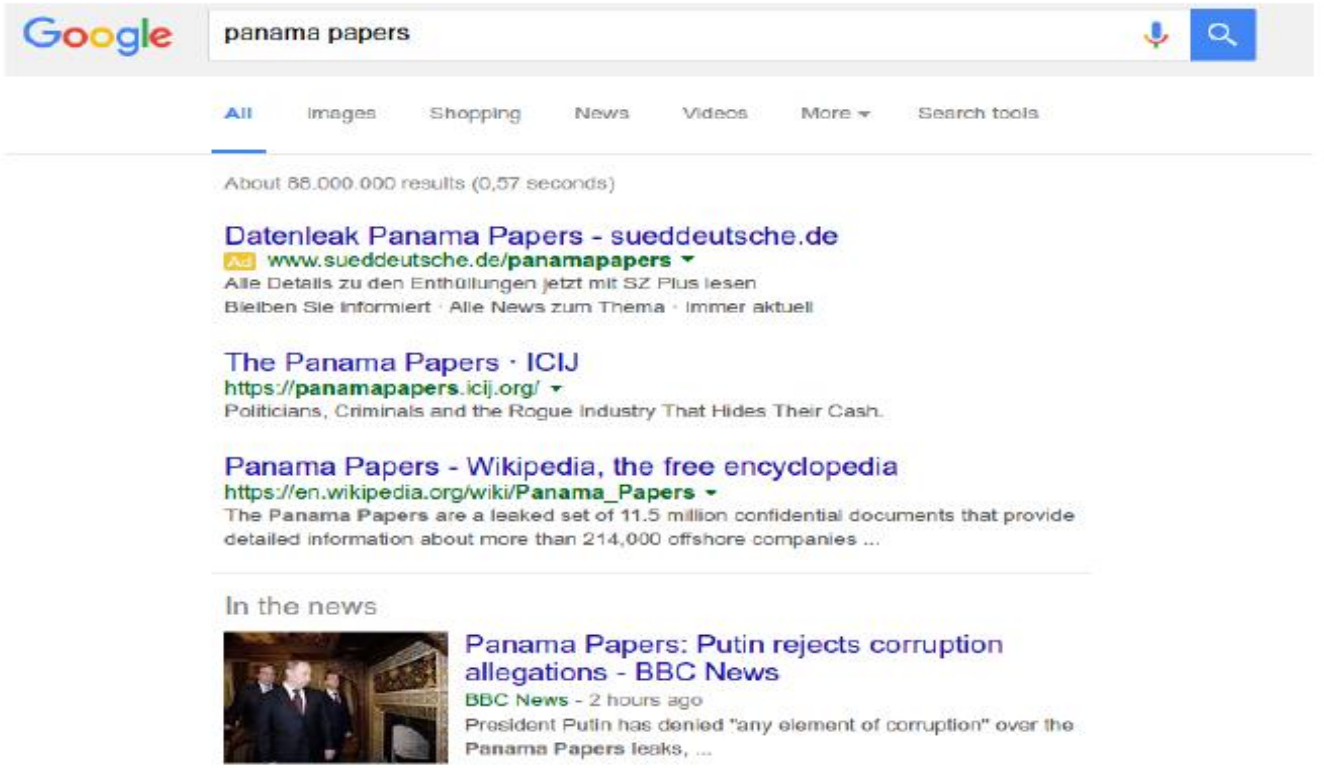
- Word prediction & Search text completion



Very challenging for personal / local / global

NLP Applications

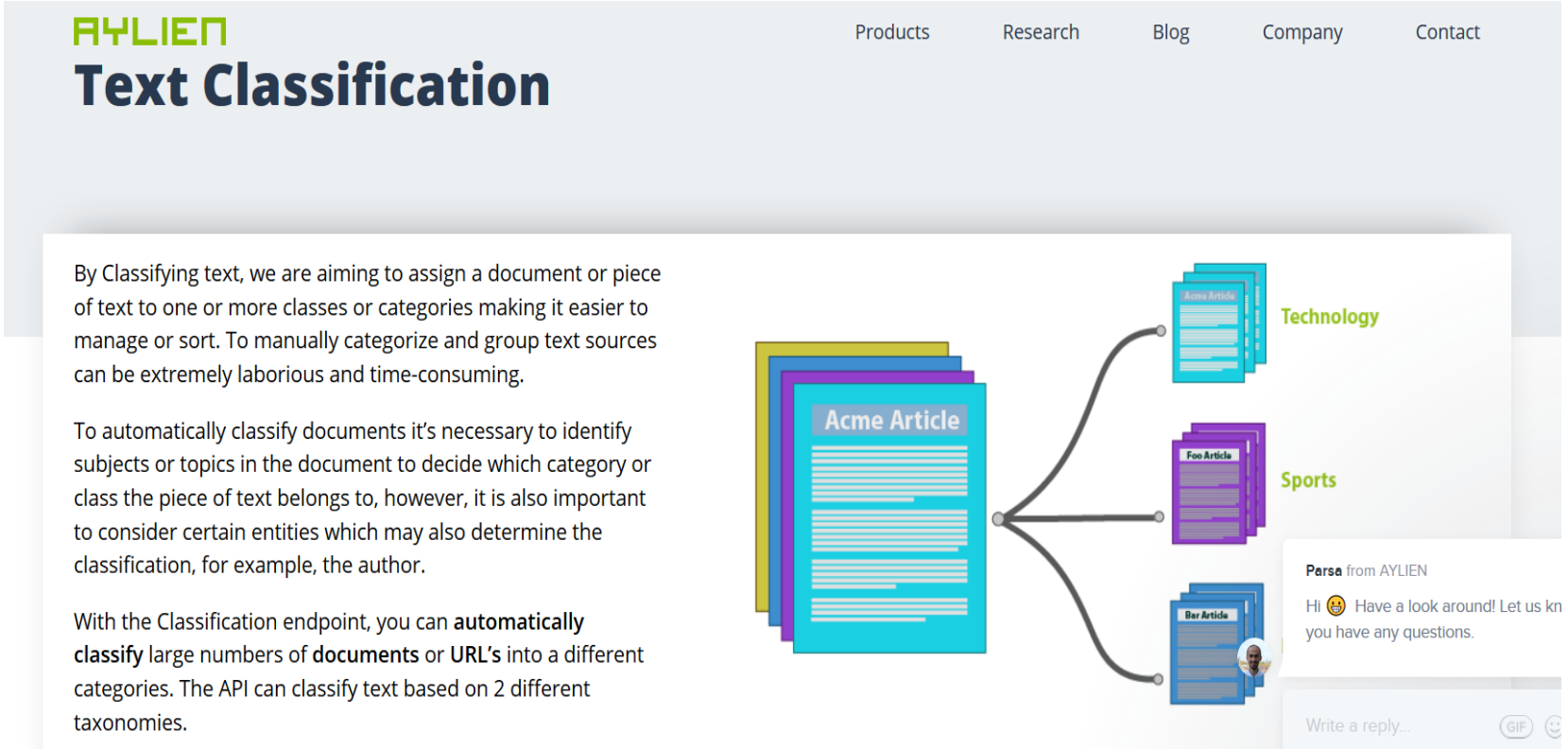
- Information Retrieval (Semantics)



Finding relevant information as per user query & Intent.

NLP Applications

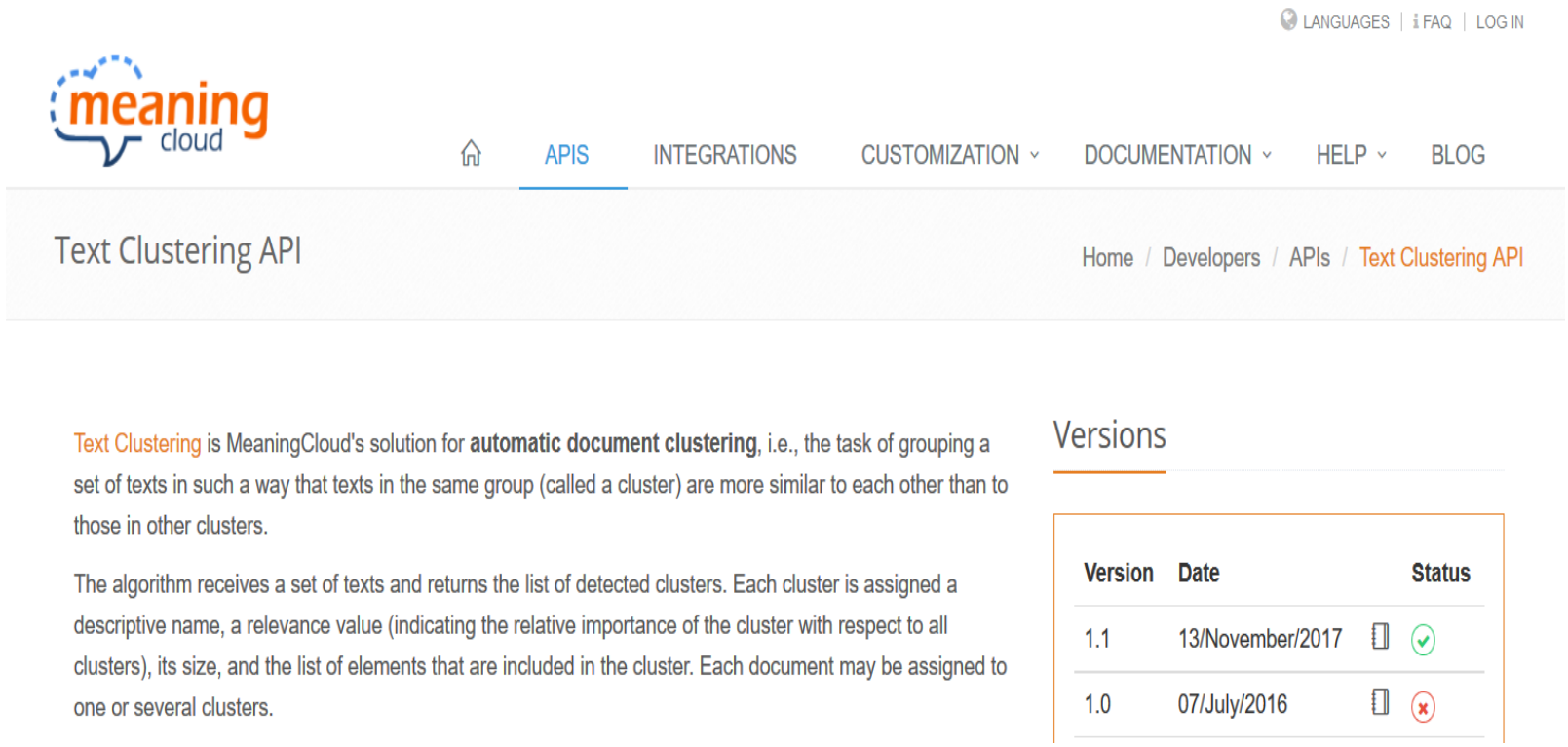
- Text Classification (Semantics)



From a predefine class assign a single class label to a given text.

NLP Applications

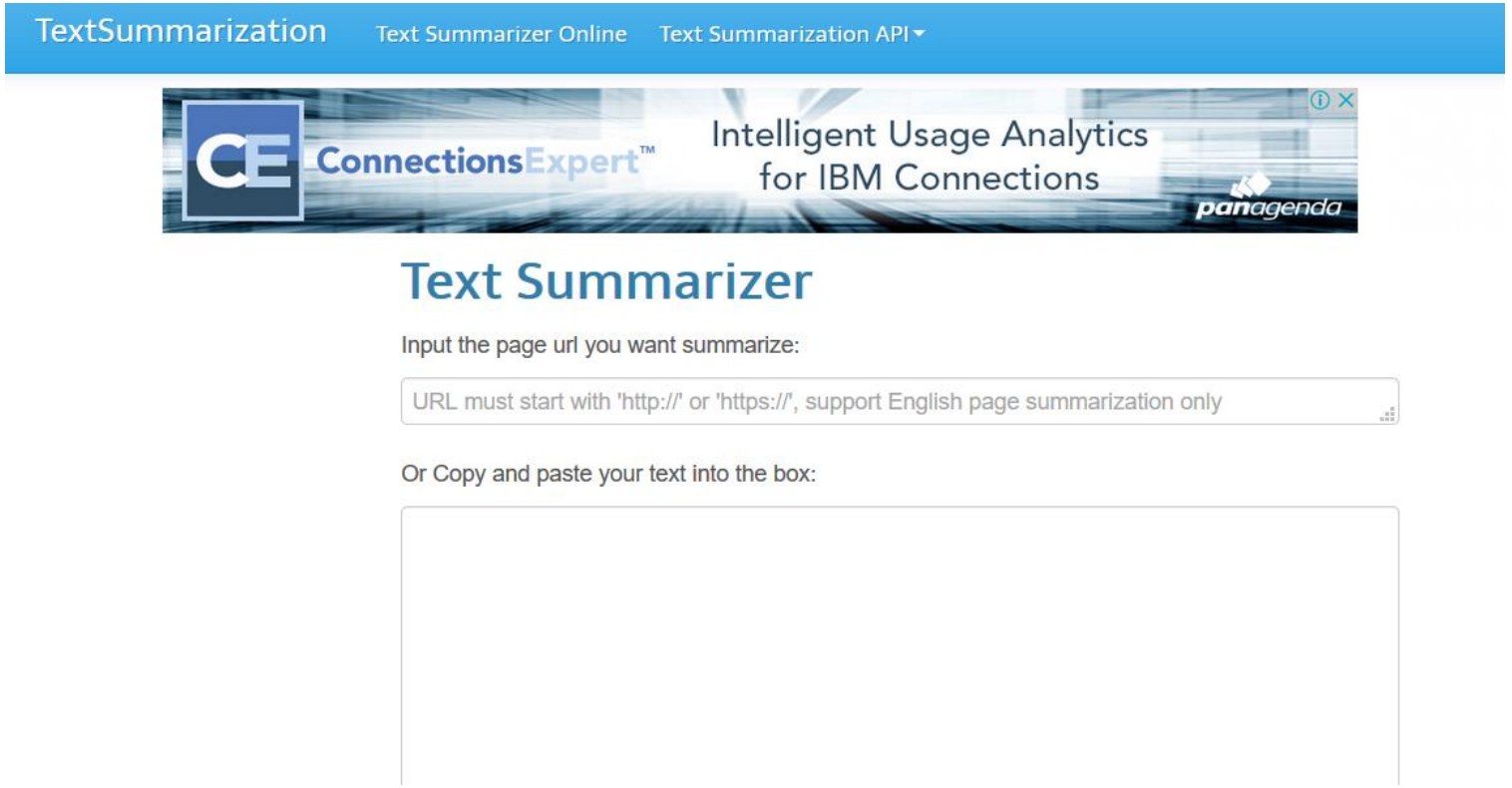
- Text Clustering (Semantics)



From a collection, identifying implicit pattern and segregate into smaller collections(clusters)- very challenging task

NLP Applications

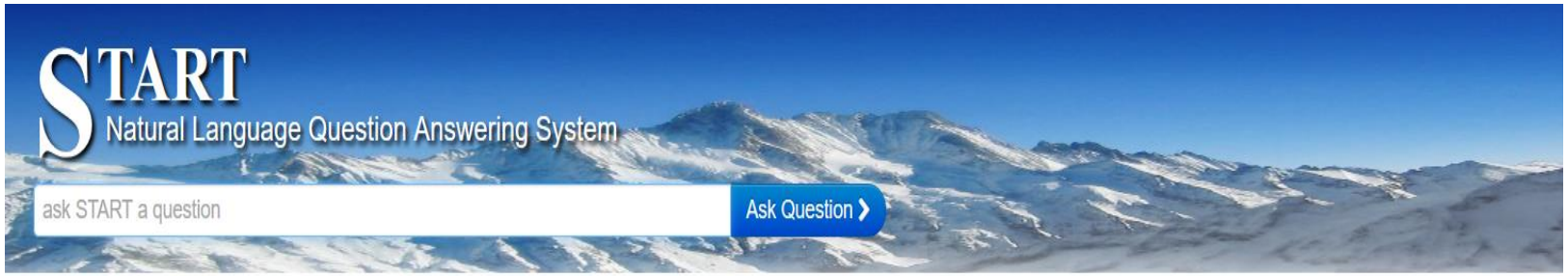
- Text Summarization (Semantics)



Given a large text collection, creating a summary from it.

NLP Applications

- Question / Answering Systems



START, the world's first Web-based question answering system, has been on-line and continuously operating since December, 1993. It has been developed by Boris Katz and his associates of the InfoLab Group at the MIT Computer Science and Artificial Intelligence Laboratory. Unlike information retrieval systems (e.g., search engines), START aims to supply users with "just the right information," instead of merely providing a list of hits. Currently, the system can answer millions of English questions about places (e.g., cities, countries, lakes, coordinates, weather, maps, demographics, political and economic systems), movies (e.g., titles, actors, directors), people (e.g., birth dates, biographies), dictionary definitions, and much, much more. Below is a list of some of the things START knows about, with example questions. You can type your question above or select from the following examples. less...

Geography

- What South-American country has the largest population?
- What's the largest city in Florida?
- Give me the states that border Colorado.
- What cities are within 250 miles of the capital of Italy?
- How many people live in Israel?

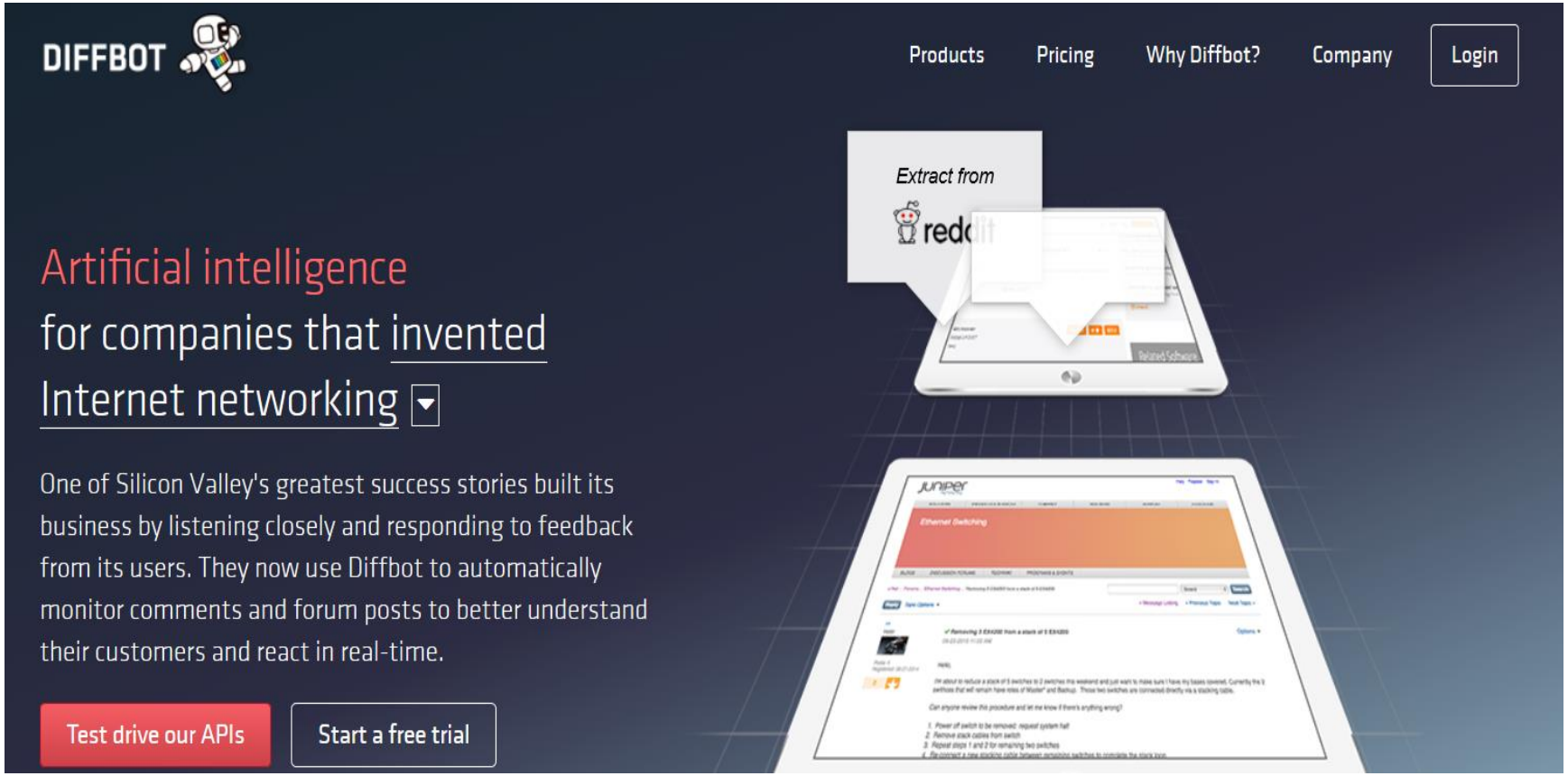
Science and Reference

- What is Jupiter's atmosphere made of?
- Who first discovered radiocarbon dating?
- How far is Neptune from the sun?
- Why is the sky blue?
- What planet has the smallest surface area?

Given a question, it will fetch a best possible answer.

NLP Applications

- Information Extraction



Get the required information from a collection/online.

NLP Applications

- Speech Recognition

Understating spoken language and answer as per the available information about the question.

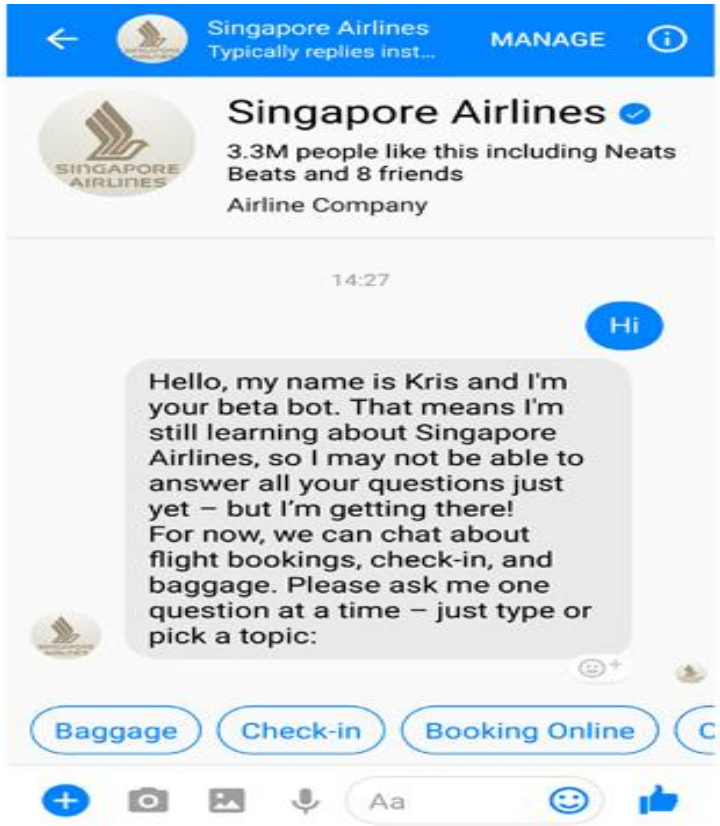
NLP Applications

- Speech Synthesis

Understating spoken/written language and answer(spoken) as per the available information about the question.

NLP Applications

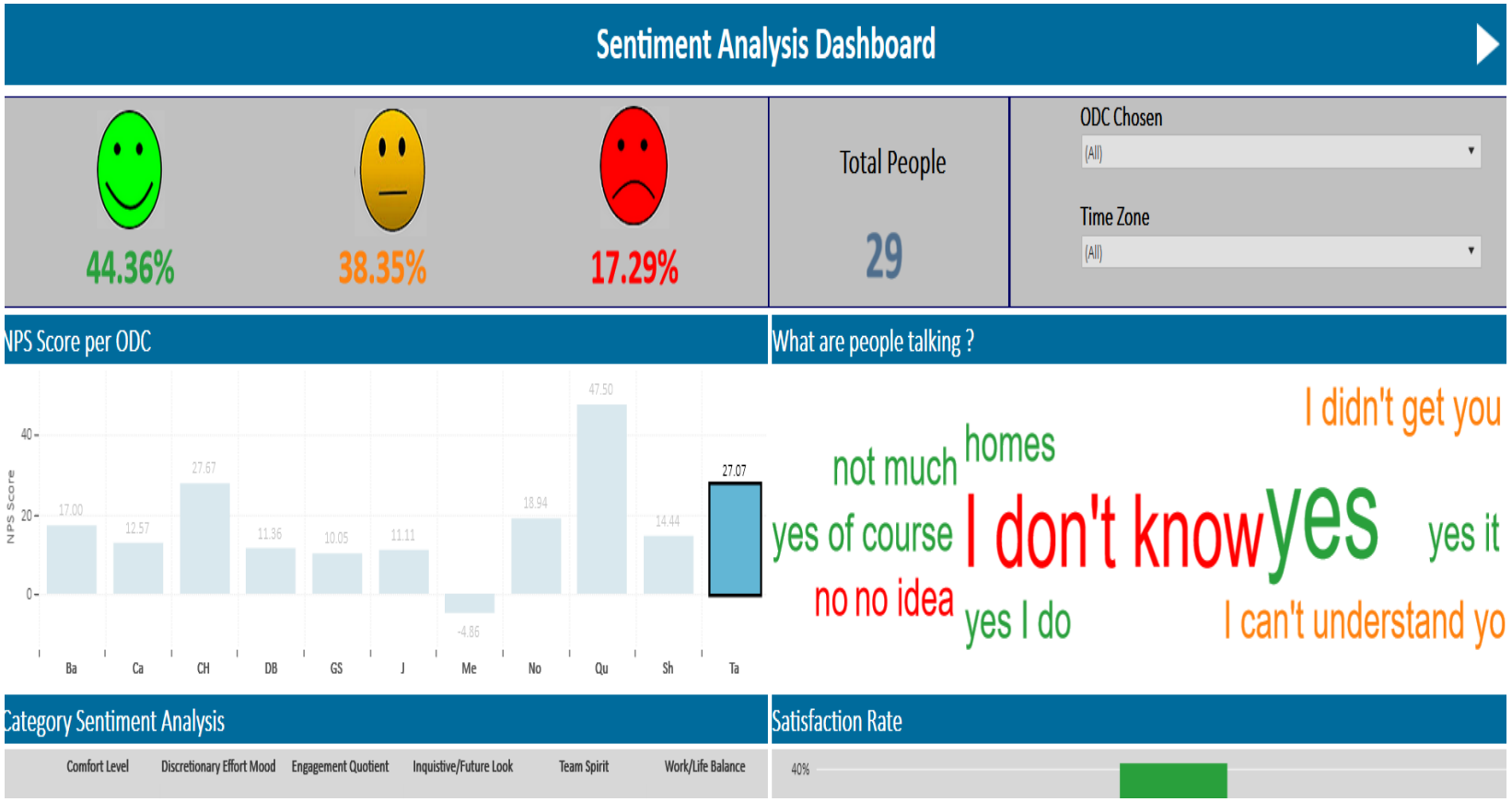
- Spoken & Written Dialog Systems (ChatBots)



Understating spoken/written language and answer(spoken) as per the available information about the question.

NLP Applications

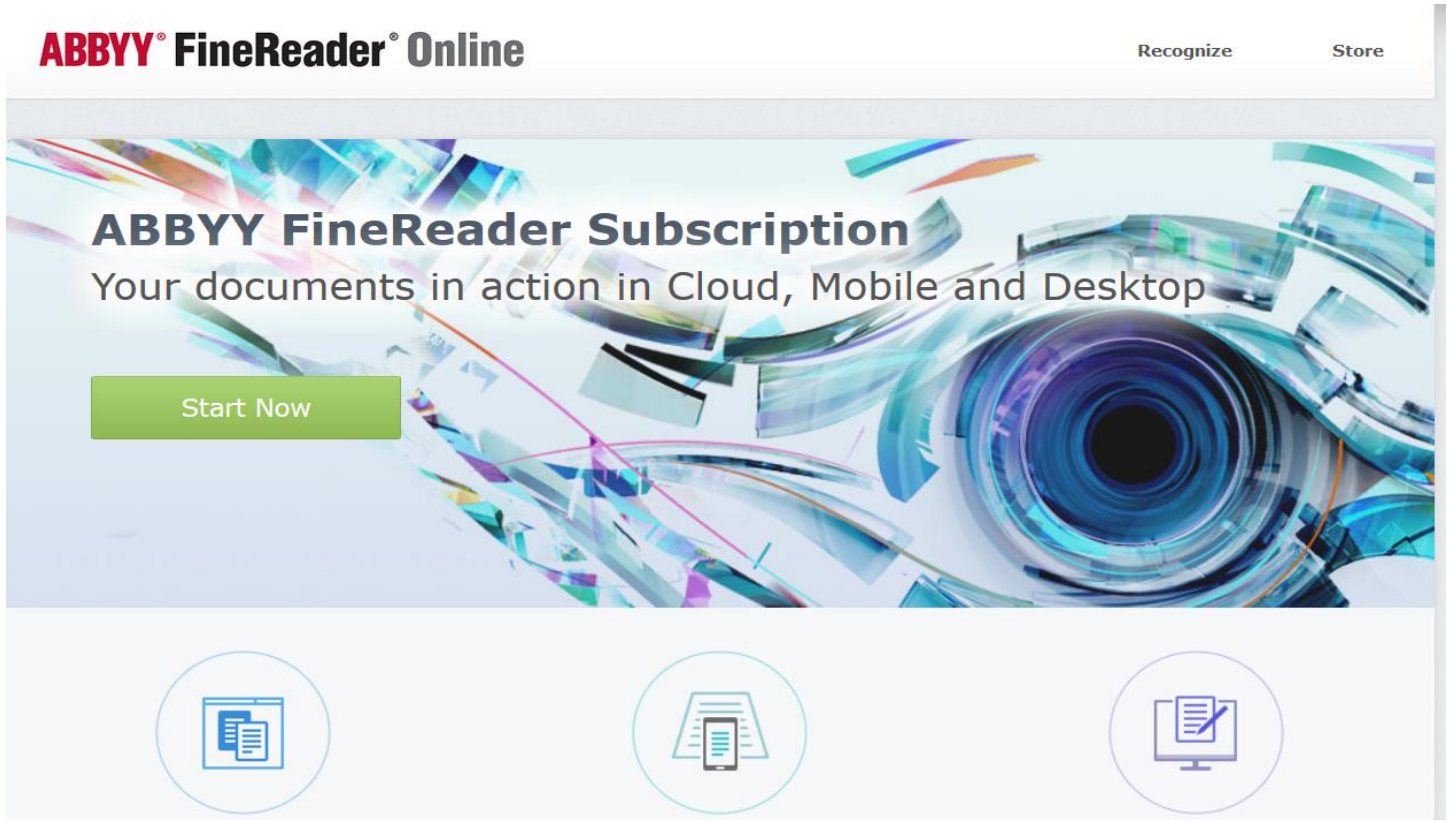
- Sentiment Analysis



Tapping the sentiment associated with some entity.

NLP Applications

- Optical Character Recognizer (OCR)



Use to identify text from image/documents.

NLP Application Areas

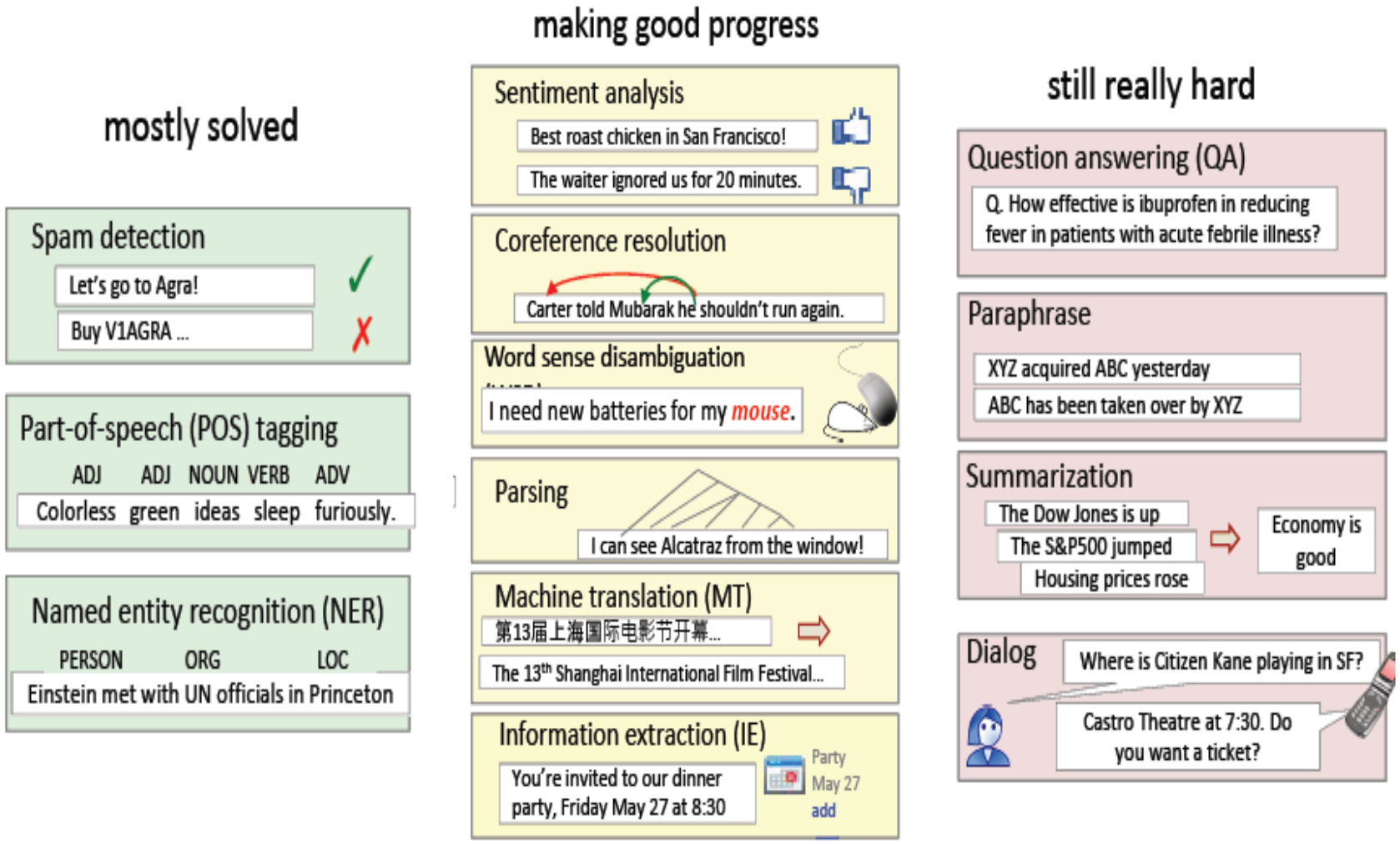
- Topic identification
- Chatbots
- Sentiment Analysis
- Translation
- Text classification

Regular Expressions are useful in all NLP application areas

Agenda

- Introduction
- Natural Language Processing
- Natural Vs. Artificial (Abstract) Languages
- Computational Linguistics
- NLP Applications Area
- NLP Tasks
- Ambiguity in NLP

NLP Tasks


mostly solved
Spam detection
Let's go to Agra!
Buy V1AGRA ...
Part-of-speech (POS) tagging
ADJ ADJ NOUN VERB ADV
Colorless green ideas sleep furiously.
Named entity recognition (NER)
PERSON ORG LOC
Einstein met with UN officials in Princeton
making good progress
Sentiment analysis
Best roast chicken in San Francisco!
The waiter ignored us for 20 minutes.
Coreference resolution
Carter told Mubarak he shouldn't run again.
Word sense disambiguation
I need new batteries for my mouse.
Parsing
I can see Alcatraz from the window!
Machine translation (MT)
第13届上海国际电影节开幕...
The 13th Shanghai International Film Festival...
Information extraction (IE)
You're invited to our dinner party, Friday May 27 at 8:30
Party
May 27
add
still really hard
Question answering (QA)
Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?
Paraphrase
XYZ acquired ABC yesterday
ABC has been taken over by XYZ
Summarization
The Dow Jones is up
The S&P500 jumped
Housing prices rose
Economy is good
Dialog
Where is Citizen Kane playing in SF?
Castro Theatre at 7:30. Do you want a ticket?

Agenda

- Introduction
- Natural Language Processing
- Natural Vs. Artificial (Abstract) Languages
- Computational Linguistics
- NLP Applications Area
- NLP Tasks
- Ambiguity in NLP

Ambiguity in NLP

- Something is ambiguous when it can be understood in two or more possible senses or ways.
- How many meanings can you get for the sentence “I saw the man on the hill with a telescope”?
  - I saw the man. The man was on the hill. I was using a telescope.
  - I saw the man. I was on the hill. I was using a telescope.
  - I saw the man. The man was on the hill. The hill had a telescope.
  - I saw the man. I was on the hill. The hill had a telescope.
  - I saw the man. The man was on the hill. I saw him using a telescope.

Ambiguity in NLP

- How many meanings can you get for the sentence “ I made her duck”
  - I cooked waterfowl for her.
  - I cooked waterfowl belonging to her.
  - I created the (plaster?) duck she owns.
  - I caused her to quickly lower her head or body.
  - I waved my magic wand and turned her into undifferentiated waterfowl.

Ambiguity in NLP

- There are generally two types of ambiguities in languages:
  - Lexical ambiguity (the presence of two or more possible meanings within a single word);
  - Syntactic ambiguity (the presence of two or more possible meanings within a single sentence or sequence of words).

Agenda

- Introduction
- Natural Language Processing
- Natural Vs. Artificial (Abstract) Languages
- Computational Linguistics
- NLP Applications Area
- NLP Tasks
- Ambiguity in NLP
- Regular Language

Regular Expression

- Regular expression is a kind of formal language for specifying text string over a character-set (language).

Natural Language Processing

- Field of study focused on making sense of language
  - Using Statistics and Computers
- Regular Expressions
  - Strings with a special syntax
  - Allow us to match patterns in other strings
  - Applications of regular expressions:
    - Find all web links in a document
    - Parse email addresses, remove/replace unwanted characters

re library python

- Provides regex functionality in Python
- Useful functions
  - re.match(*pattern*,*string*) – returns a match object
    - Match always start pattern matching from the start of the string
  - re.split() – split a string on regex
  - findall() – find all patterns in a string
  - search() – search for a pattern that appears anywhere in the string

Python's re Module

- `re` module
- `split`: split a string on regex
- `findall`: find all patterns in a string
- `search`: search for a pattern
- `match`: match an entire string or substring based on a pattern

- Pattern first, and the string second
- May return an iterator, string, or match object

```
In [5]: re.split('\s+', 'Split on spaces.')
Out[5]: ['Split', 'on', 'spaces.']
```

Which pattern?

Which of the following Regex patterns results in the following text?

```
>>> my_string = "Let's write RegEx!"
>>> re.findall(PATTERN, my_string)
['Let', 's', 'write', 'RegEx']
```

Possible Answers

- PATTERN = r"\s+"
- PATTERN = r"\w+"
- PATTERN = r"[a-z]"
- PATTERN = r"\w"

More regex practice

- Difference between `re.search()` and `re.match()`

```
In [1]: import re

In [2]: re.match('abc', 'abcde')
Out[2]: <_sre.SRE_Match object; span=(0, 3), match='abc'>

In [3]: re.search('abc', 'abcde')
Out[3]: <_sre.SRE_Match object; span=(0, 3), match='abc'>

In [4]: re.match('cd', 'abcde')

In [5]: re.search('cd', 'abcde')
Out[5]: <_sre.SRE_Match object; span=(2, 4), match='cd'>
```

What is tokenization?

- Turning a string or document into **tokens** (smaller chunks)
- One step in preparing a text for NLP
- Many different theories and rules
- You can create your own rules using regular expressions
- Some examples:
    - Breaking out words or sentences
    - Separating punctuation
    - Separating all hashtags in a tweet

nltk library

- `nltk`: natural language toolkit

```
In [1]: from nltk.tokenize import word_tokenize

In [2]: word_tokenize("Hi there!")
Out[2]: ['Hi', 'there', '!']
```

Why tokenize?

- Easier to map part of speech
- Matching common words
- Removing unwanted tokens
- "I don't like Sam's shoes."
- "I", "do", "n't", "like", "Sam", "'s", "shoes", "."

Other nltk tokenizers

- `sent_tokenize`: tokenize a document into sentences
- `regexp_tokenize`: tokenize a string or document based on a regular expression pattern
- `TweetTokenizer`: special class just for tweet tokenization, allowing you to separate hashtags, mentions and lots of exclamation points!!!

Regex groups using or "|"

- OR is represented using `|`
- You can define a group using `()`
- You can define explicit character ranges using `[]`

```
In [1]: import re

In [2]: match_digits_and_words = ('(\d+|\w+)')

In [3]: re.findall(match_digits_and_words, 'He has 11 cats.')
Out[3]: ['He', 'has', '11', 'cats']
```

Regex ranges and groups

pattern	matches	example
[A-Za-z]+	upper and lowercase English alphabet	'ABCDEFghijk'
[0-9]	numbers from 0 to 9	9
[A-Za-z\-\.]+	upper and lowercase English alphabet, - and .	'My-Website.com'
(a-z)	a, - and z	'a-z'
(\s+l,)	spaces or a comma	', '

Character range with re.match()

```
In [1]: import re

In [2]: my_str = 'match lowercase spaces nums like 12, but no commas'

In [3]: re.match('[a-z0-9 ]+', my_str)
Out[3]: <_sre.SRE_Match object;
          span=(0, 42), match='match lowercase spaces nums like 12'>
```

Bag-of-words

- Basic method for finding topics in a text
- Need to first create tokens using tokenization
- ... and then count up all the tokens
- The more frequent a word, the more important it might be
- Can be a great way to determine the significant words in a text

Bag-of-words example

- Text: "The cat is in the box. The cat likes the box. The box is over the cat."
- Bag of words (stripped punctuation):
  - "The": 3, "box": 3
  - "cat": 3, "the": 3
  - "is": 2
  - "in": 1, "likes": 1, "over": 1

Bag-of-words in Python

```
In [1]: from nltk.tokenize import word_tokenize

In [2]: from collections import Counter

In [3]: Counter(word_tokenize(
                """The cat is in the box. The cat likes the box.
                 The box is over the cat."""))
Out[3]:
Counter({'.': 3,
         'The': 3,
         'box': 3,
         'cat': 3,
         'in': 1,
         ...
         'the': 3})

In [4]: counter.most_common(2)
Out[4]: [('The', 3), ('box', 3)]
```

Regex Characters

Char	*Legend*	*Example*	*Match*
\d	one Unicode digit in any script	file_\d\d	file_9੩
\w	"word character": Unicode letter, ideogram, digit, or underscore	\w-\w\w\w	c-k_u
\s	"whitespace character": any Unicode separator	a\sb\sc	a b c
\D	One character that is not a *digit*	\D\D\D	ABC
\W	One character that is not a *word character* as defined by your engine's *\w*	\W\W\W\W\W	*-+=)
\S	One character that is not a *whitespace character*	\S\S\S\S	Yoyo
[a-z]	lower case group	B[a-c]D	BaD
.	wildcard	B..D	Bq9D

Regex Quantifiers

Quantifier	Legend	Example	Sample Match
+	One or more	Hi \w-\w+	Hi A-b1_1
{3}	Exactly three times	\D{3}	ABC
{2,4}	Two to four times	\d{2,4}	156
{3,}	Three or more times	\w{3,}	regex_tutorial
*	Zero or more times	A*B*C*	AAACC
?	Once or none	plurals?	plural

Character Classes

Char	Legend	example	Match
[]	one of the characters in the brackets	B**[AEIOU]**D	BUD
[x-y]	one character in the range	B**[a-c3-9W-Z]**D	B3D
[^x]	one character that is not x	B[^c^k]D	BOD
[^x-y]	one character not in range	B[^a-z]D	BAD
[\x41]	matches character with specified ASCII code		
[\U00032]	matches character with specified Unicode code		

Regex Logical Operators

Logic	Legend	Example	Sample Match
|	OR operand	22|33	33
(…)	Capturing group	A(nt|pple)	Apple
\1	Contents of Group 1	r(\w)g\1x	regex
\2	Contents of Group 2	(\d\d)\+(\d\d)=\2 \+\1	12+65=65+12
(?: …)	Non-capturing group	A(?:nt|pple)	Apple

Tokenization

- Transforming a string or documents into smaller parts called **Tokens**, e.g.
  - Extracting words
  - Extracting numbers
  - Separate all hashtags in a tweet
- An important pre-processing step in NLP
- Python Natural Language Toolkit library (nltk) provides strong support for most NLP tasks

Tokenization

- Sample Python code for tokenization:
  (see Jupyter notebook)
- Tokenization application
  - Part of speech tagging
  - removing unwanted tokens, such as stop words
  - Text minning
  - Information retrieval

Tokenizers in nltk

- Several tokenizers are available in nltk
- Most common tokenizers are
  - **word_tokenize**: generates word-by-word tokens
  - **sent_tokenize**: tokenizes a document into sentences
  - **regexp_tokenize**: generates tokens according to a regular expression
  - **TweetTokenizer**: tokenize tweets

Simple NLP Tasks

Bag of Words

- A simple method to find Topics in a text
  - Generate tokens
  - Convert all tokens to lower case
  - Count frequency of each token in the text
  - The characters with highest frequency are the most significant words

Simple Preprocessing Steps

- Convert text into strings
- Changing tokens to lowercase
- Removing numbers and punctuation signs
- Removing stop words
  - A built-in library has been provided in Python
- Lemmatization/stemming: Shorten words to their root stems, i.e. converting plurals to signgular forms

Gensim

- Widely used open source vector-space modeling and topic modeling library
- Classes and functions to easily realise complex NLP tasks
- It is built on top of NumPy and SciPy
- Gensim includes implementations of tf-idf, random projections, word2vec and document2vec algorithms, hierarchical Dirichlet processes (HDP), latent semantic analysis (LSA) and latent Dirichlet allocation (LDA), including distributed parallel versions.