Università degli Studi di Catania

Dipartimento di Matematica e Informatica

Sentiment Analysis

Alessandro Ortis, PhD
ortis@dmi.unict.it

IMAGE PROCESSING LABORATORY
http//iplab.dmi.unict.it

Outline

- Sentiment Analysis
    - Introduction
    - Natural Language Processing (NLP)
    - Formalization
    - Use of Machine Learning
    - Lexical Resources

Introduction

Sentiment Analysis / Opinion Mining

"Sentiment analysis is the computational study of people's opinions, sentiments, emotions, and attitudes. This fascinating problem is increasingly important in business and society. It offers numerous research challenges but promises insight useful to anyone interested in opinion analysis and social media analysis."

"Sentiment Analysis: mining sentiments, opinions, and emotions"
Bing Liu
Cambridge University Press, June 2015

Introduction

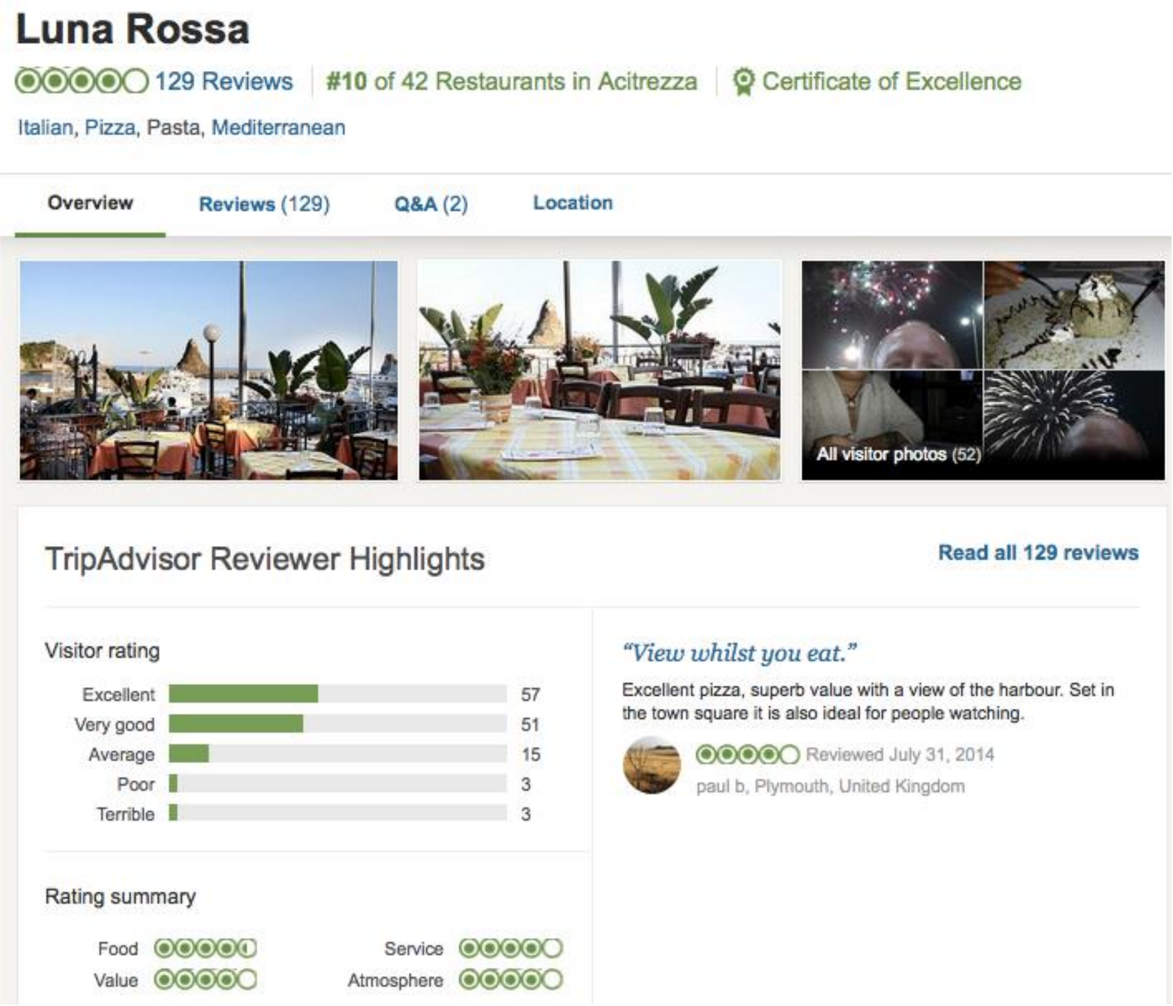
Sentiment Analysis / Opinion Mining

SENTIMENT ANALYSIS

Discovering people opinions, emotions and feelings about a product or service

Introduction

Motivations



Introduction

Motivations

Nikon D90 12.3 MP Digital SLR Camera - AF-S DX 18-105mm Lens

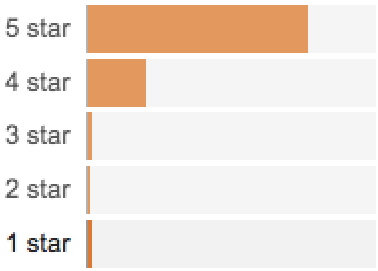
$549 online

★★★★★ 1,000 product reviews | **Save to Shortlist** | Browse Digital Cameras »

September 2009 · Nikon · Nikon D Series · DSLR · 12.3 megapixel · Crop Sensor · CMOS · Built-in Flash · Detachable Flash · 22.4 ounce

« Back to overview

Reviews



4.7
★★★★★
1,000 reviews

pictures	"The camera is very durable and takes great pictures."
value	"Wonderful camera at a wonderful price."
zoom/lens	"Overall a Great DSLR."
features	"Love the "Raw+JPEG fine" mode."
design	"A nice camera, with good ergonomics."
screen	"LCD is larger more refined, nice to have live view."
battery life	"Battery life is good."

Showing 1 star reviews - Show all reviews

Introduction

Motivations

Social Media Monitoring: monitoring conversations happening on social media channels about your brand/company.

Introduction

Motivations

Rom in gabbia, il popolo di Facebook contro Lidl: "Non licenziate i due dipendenti"

CRONACA

Mi piace 17 mila | Condividi | Tweet | Condividi

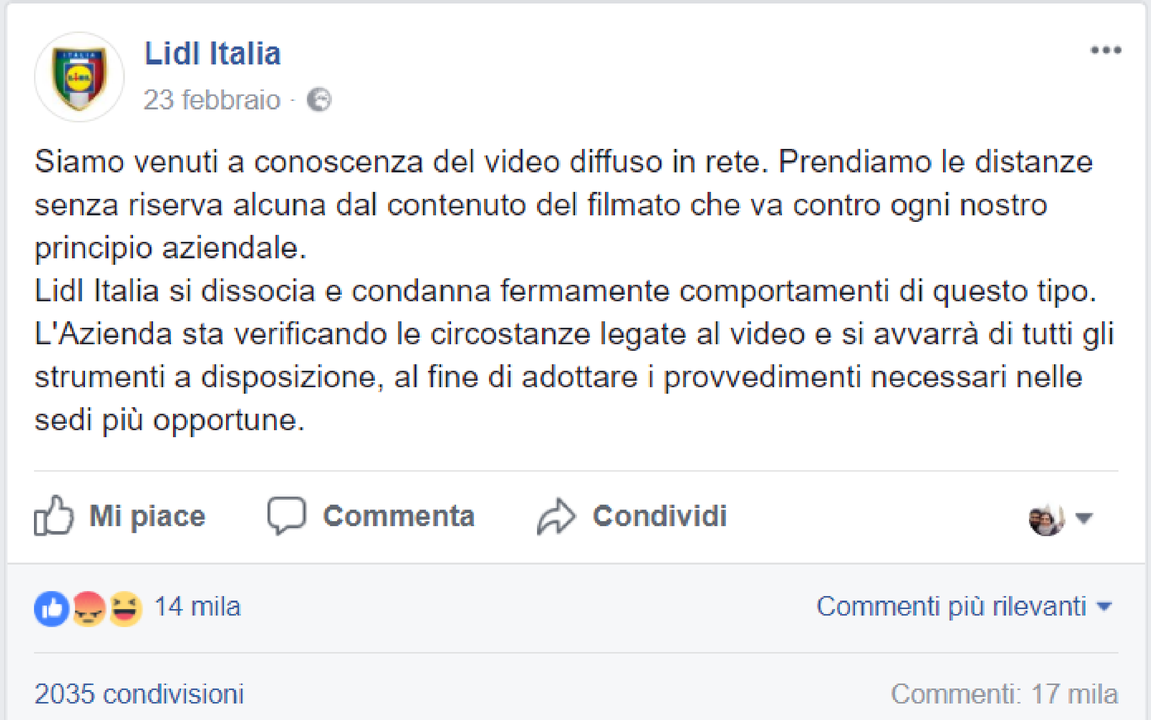
Pubblicato il: 24/02/2017 11:24

"Se licenziate i due ragazzi **io e la mia famiglia non metteremo più piede nella vostra catena**. Italia agli italiani! Bravi i due ragazzi!". E' solo il primo di una lunga sequela di commenti che è possibile leggere sotto il post di Facebook con cui la catena di supermercati **Lidl Italia** ha preso le distanze dal video girato da due dipendenti di Follonica che giovedì scorso hanno rinchiuso due donne rom in una gabbia adibita alla raccolta rifiuti, mentre erano intente a rovistare tra la spazzatura.

Una decisione poco gradita al popolo del social network, che si è subito schierato dalla parte dei due uomini.

Facebook Post ID: 421326344567984_1473525486014726

Introduction


Lidl Italia

23 febbraio ·

Siamo venuti a conoscenza del video diffuso in rete. Prendiamo le distanze senza riserva alcuna dal contenuto del filmato che va contro ogni nostro principio aziendale.
Lidl Italia si dissocia e condanna fermamente comportamenti di questo tipo.
L'Azienda sta verificando le circostanze legate al video e si avvarrà di tutti gli strumenti a disposizione, al fine di adottare i provvedimenti necessari nelle sedi più opportune.

Mi piace Commenta Condividi

14 mila Commenti più rilevanti

2035 condivisioni Commenti: 17 mila

Facebook Post ID: [421326344567984_1473525486014726]

Introduction

Facebook Post ID: [421326344567984_1473525486014726](421326344567984_1473525486014726)

Introduction

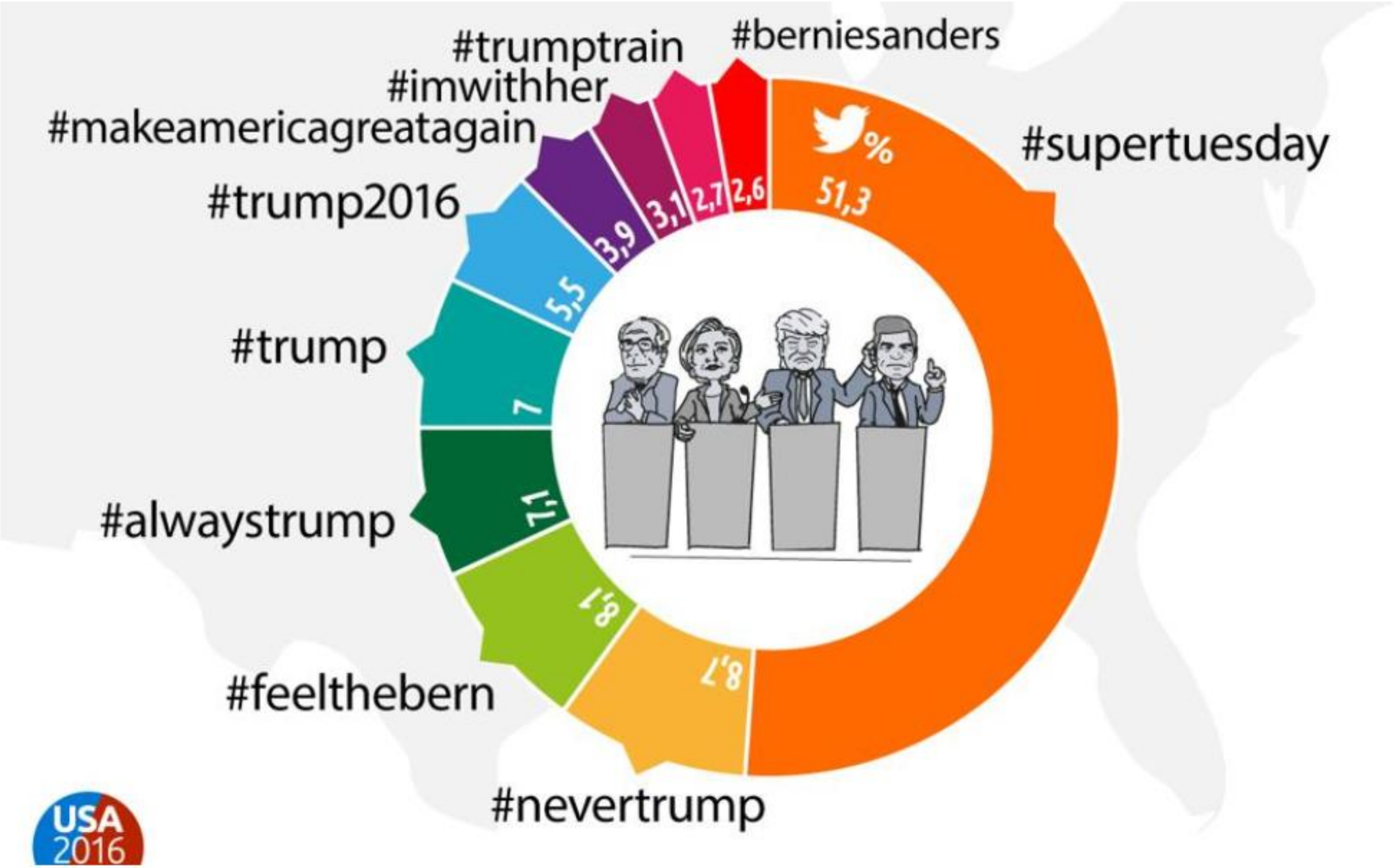
Motivations

Negative comments about the brand

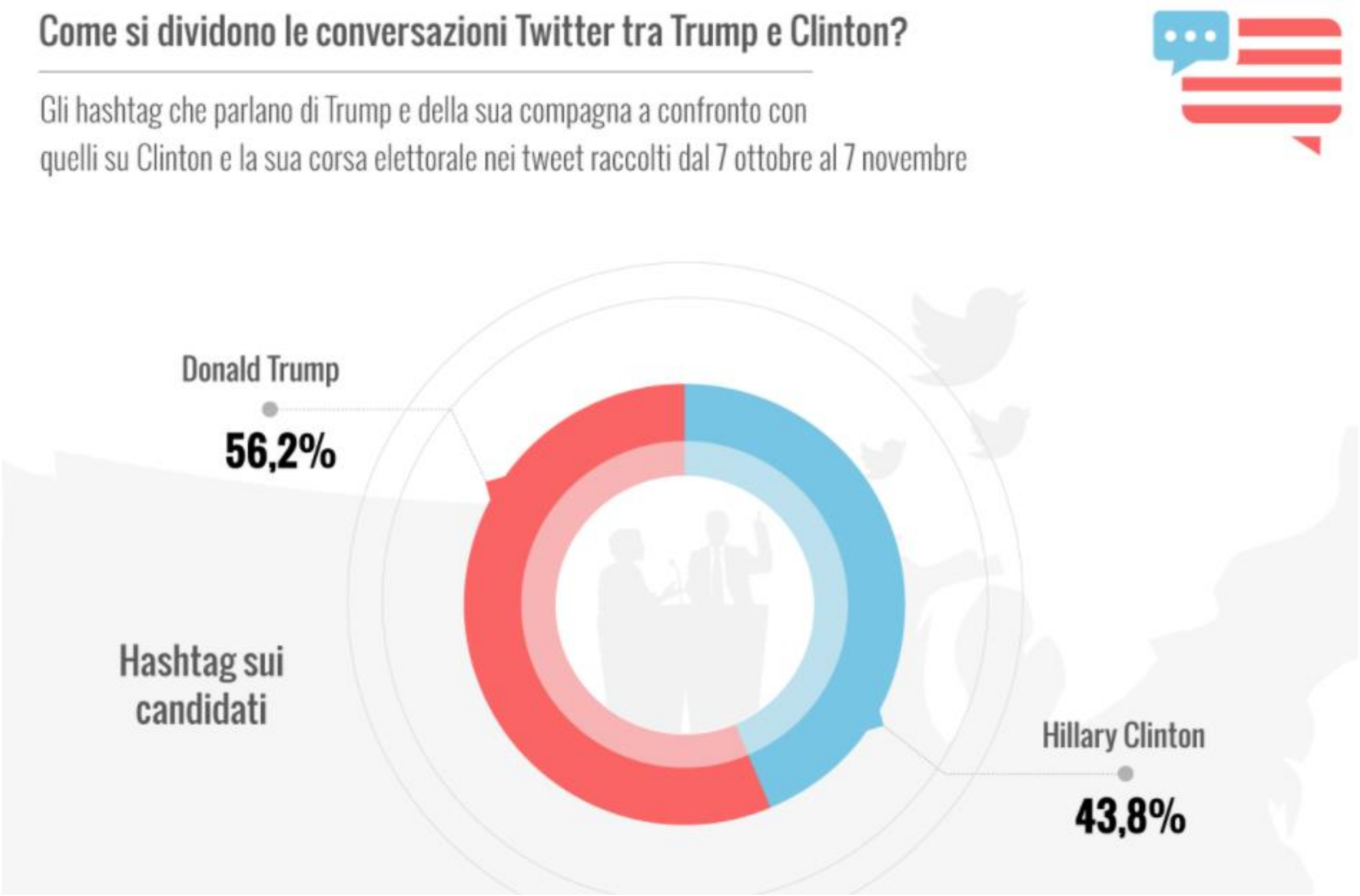

```
{
  "created_time": "2017-02-24T11:24:38+0000",
  "message": "Ma da cosa prendete le distanze?? E quali provvedimenti andate minacciando?? Tute
  "id": "1473525486014726_924061564402248"
},
{
  "created_time": "2017-02-24T11:46:30+0000",
  "message": "Io mi dissocio da tali idioti. A mio avviso gente come MS e i cerebrolesi che sos
  "id": "1473525486014726_402549403442860"
},
{
  "created_time": "2017-02-23T23:12:18+0000",
  "message": "Io mi auguro che vengano licenziati. Nessuno deve sostituirsi allo stato.  Hanno
  "id": "1473525486014726_623577094496862"
},
{
  "created_time": "2017-02-24T17:10:02+0000",
  "message": "Dopo questo comunicato perderete molti clienti. Dovreste premiare le vittime che
  "id": "1473525486014726_622029997970521"
},
{
  "created_time": "2017-02-24T12:15:58+0000",
  "message": "Cara lidl con queste affermazioni volete difendere i ladri che rubano dentro e fu
  "id": "1473525486014726_395390230819871"
},
{
  "created_time": "2017-02-24T05:52:31+0000",
  "message": "Io sto dalla parte dei ragazzi. Non hanno arrecato violenza fisica.
Sarebbe molto grave se perdessero il lavoro per questo visto che il loro gesto salvaguarda la vostra
Le urla stesse delle donne sono pesantemente esagerate in relazione al fatto che fossero li per rubar
```

Introduction

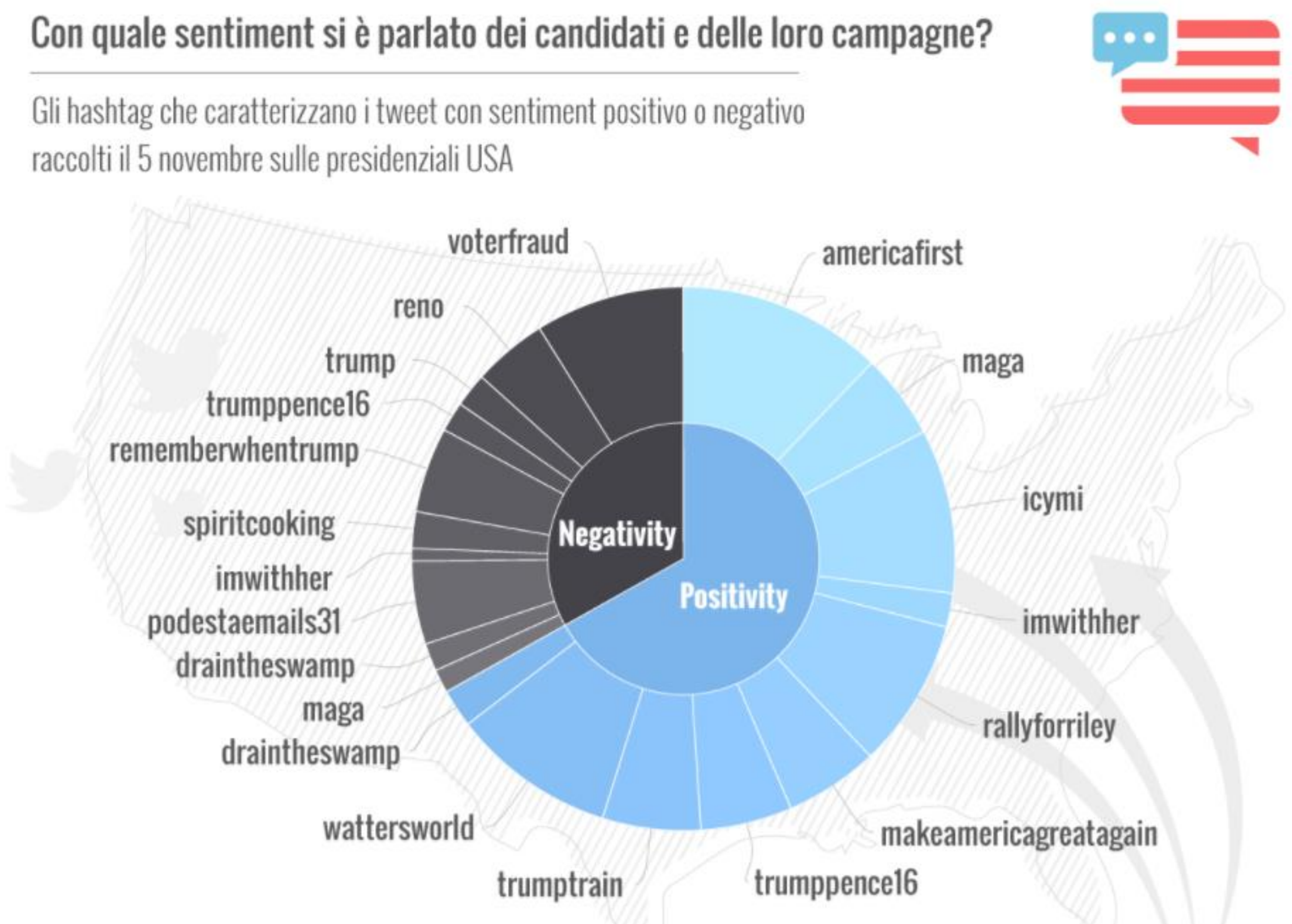
Motivations



Introduction
Motivations

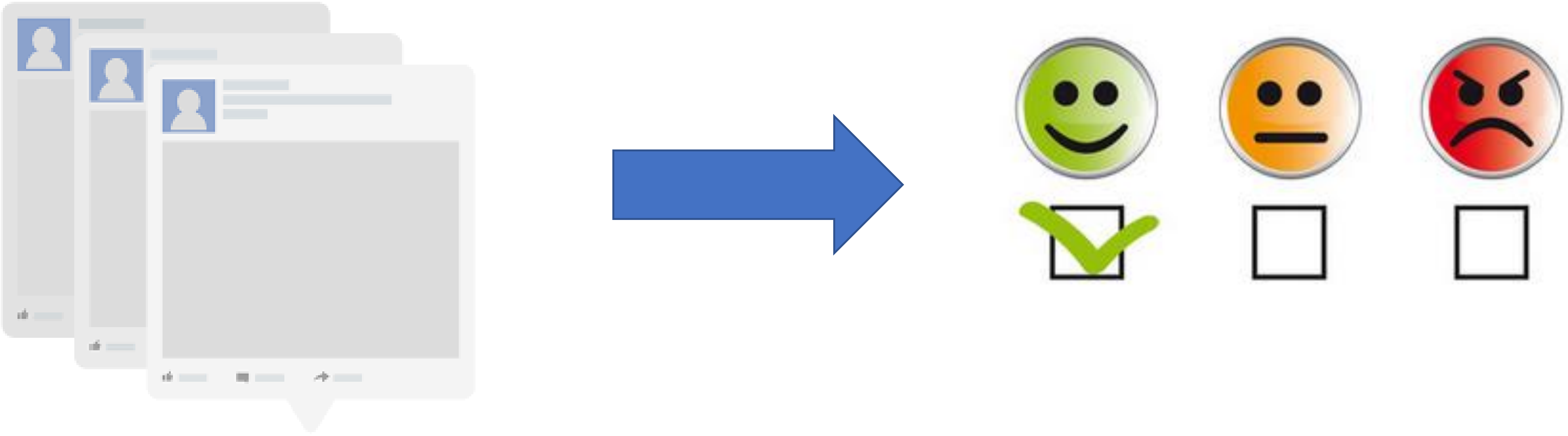


Introduction
Motivations



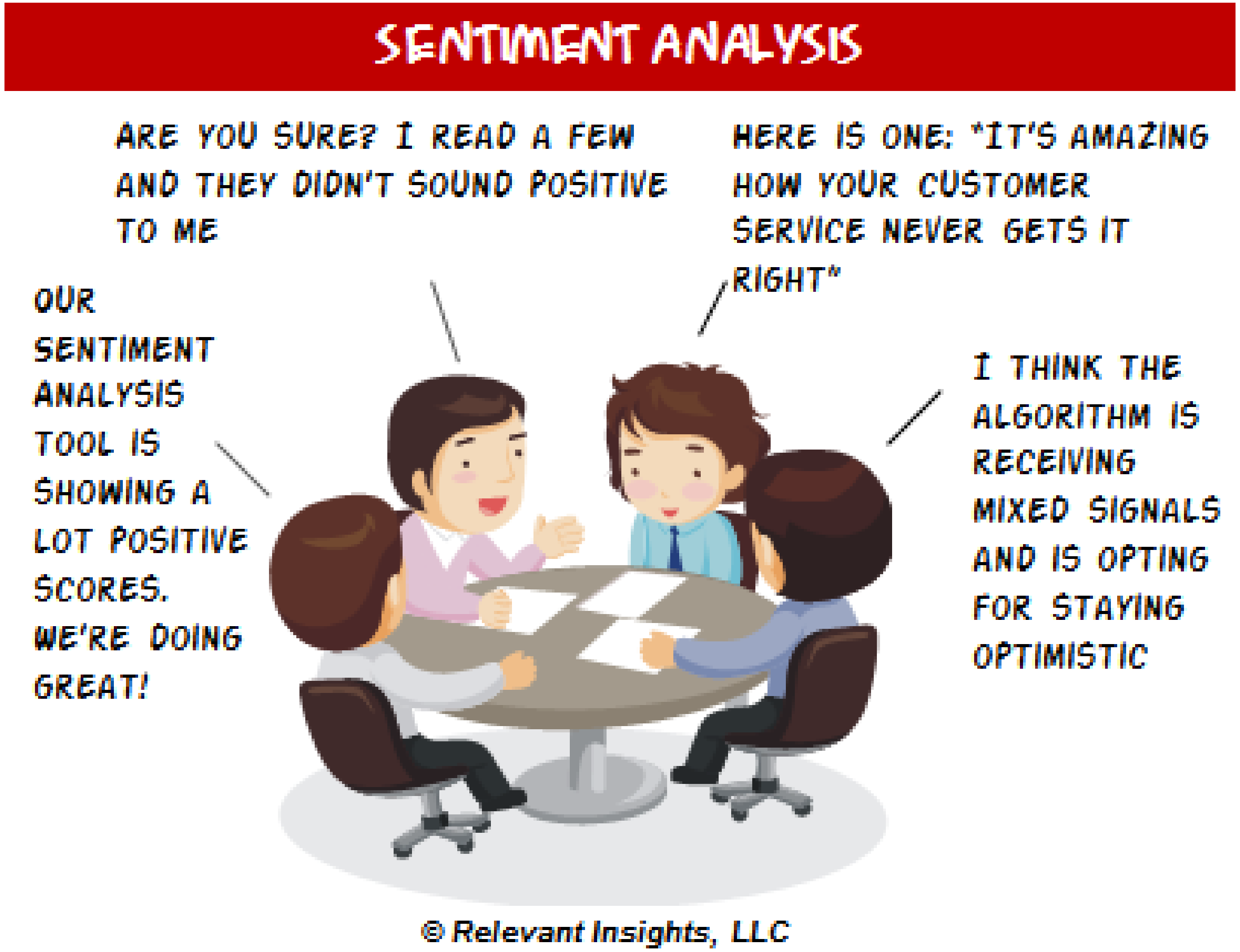
Introduction

What is Sentiment Analysis of Social Posts?



Introduction

Problems



Introduction

How to perform Sentiment Analysis?

Introduction

How to perform Sentiment Analysis?

NLP

NLP

NLP

NLP

Machine Learning
&
Pattern Recognition

Introduction

Natural Language Processing (NLP)

- Sentence Segmentation
- Part-of-speech tagging
- Stemming&Lemmatizing
- Stop words removal

Machine Learning
&
Pattern Recognition

- Data Representation
- Sentiment Classification

Natural Language Processing

Tokenization: In Natural Language Processing (NLP), tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. Tokenization is useful both in linguistics and in computer science, where it forms part of lexical analysis.

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
```

Natural Language Processing

Part of Speech (POS): in corpus linguistics, part-of-speech tagging (POS tagging), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph.

```
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Natural Language Processing

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Natural Language Processing

Stemming and Lemmatizing: In Natural Language Processing (NLP), the goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

am, are, is → *be*
car, cars, car's, cars' → *car*
the boy's cars are different colors → *the boy car be differ color*

Stemming usually refers to a crude heuristic process that chops off the ends of words. ***Lemmatization*** refers to doing things properly with the use of a vocabulary aiming to return the base or dictionary form of a word, which is known as the *lemma.*

Natural Language Processing

Stop words removal: stop word is a commonly used word (such as "the", "a", "an", "in") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We can remove them easily, by storing a list of words that you consider to be stop words.

NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

Natural Language Processing

```python
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = "This is a sample sentence, showing off the stop words filtration."

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens if not w in stop_words]

print(word_tokens)
print(filtered_sentence)

```

Output

```
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing',
'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop',
'words', 'filtration', '.']
```

Tasks

Simplest task
– Is the attitude of the text positive or negative?

More complex
– Rank the attitude of the text from 1 to 5

Advanced
– Detect the target, holder, or complex attitude types

Simple algorithm for polarity detection
(step-by-step example)

- Preprocess the text (e.g. tokenize, split in sentences, and POS tags)
- Definition of dictionary of positive and negative expressions
- Tagging tokens with dictionaries
- Measure the sentiment

Simple algorithm for polarity detection

Input text:

”What can I say about this place. The staff of the restaurant is nice and the eggplant is not bad.
Apart from that, very uninspired food, lack of atmosphere and too expensive. I am a staunch vegetarian and was sorely disappointed with the veggie options on the menu. Will be the last time I visit, I recommend others to avoid.”

Simple algorithm for polarity detection

Breaking the text in sentences:

What can I say about this place.

The staff of the restaurant is nice and the eggplant is not bad.

Apart from that, very uninspired food, lack of atmosphere and too expensive.

I am a staunch vegetarian and was sorely disappointed with the veggie options on the menu.

Will be the last time I visit, I recommend others to avoid.

Simple algorithm for polarity detection

Sentiment dictionaries:

positive.yml
nice: [positive]
awesome: [positive]
cool: [positive]
superb: [positive]

negative.yml
bad: [negative]
uninspired: [negative]
expensive: [negative]
dissapointed: [negative]
avoid: [negative]

Simple algorithm for polarity detection

Apply dictionaries to detect positive/negative words:

What can I say about this place.

The staff of the restaurant is nice and the eggplant is not bad.

Apart from that, very uninspired food, lack of atmosphere and too expensive.

I am a staunch vegetarian and was sorely disappointed with the veggie options on the menu.

Will be the last time I visit, I recommend others to avoid.

Simple algorithm for polarity detection

Sentiment measure

- Simply counting how many positive and negative expressions we detected, could be a (very naive) sentiment measure.
- Sentiment measure = -4, as there are 5 negative terms and 1 positive

(Nice +1) + (Bad -1) + (Uninspired -1) + (Expensive -1) + (Disappointed -1) + (Avoid -1)

Simple algorithm for polarity detection

Incrementers and decrementers

The previous “sentiment score” was very basic: it only counts positive and negative expressions and makes a sum, without taking into account that maybe *some expressions are more positive or more negative than others*.

inc.yml
too: [inc]
very: [inc]
sorely: [inc]

dec.yml
barely: [dec]
little: [dec]

Simple algorithm for polarity detection

Updating Example

very uninspired	(**'very**', 'very', ['inc', 'RB']), ('uninspired’, 'uninspire', ['negative', 'VBN']),
too expensive	((**'too**', 'too', ['inc', 'RB']), ('expensive', 'expensive', ['negative', 'JJ']),
sorely disappointed	(**'sorely**', 'sorely', ['inc', 'RB']), ('dissapointed', 'dissapoint', ['negative','VBN']),

Simple algorithm for polarity detection

New sentiment measure

Now, we could improve in some way our sentiment score. The idea is that "good" has more strength than "barely good" but less than "very good".

New score is (Nice +1) + (Bad -1) + (Very uninspired -2) + (Too expensive -2) + (Sorely disappointed -2) + (Avoid -1) = -7

Notice that the review is now considered more negative, due to the appearance of expressions such as "very uninspired", "too expensive" and "sorely dissapointed”.

Simple algorithm for polarity detection

Inverters and polarity flips

With the approach we've been following so far, some expressions could be incorrectly tagged. For example, this part of our example review:

*the eggplant is **not** bad*

contains the word *bad* but the sentence is a positive opinion about the eggplant. This is because the appearance of the negation word ***not***, that flips the meaning of the negative adjective *bad*. We could take into account these types of polarity flips defining a dictionary of inverters:

inv.yml

lack of: [inv]
not: [inv]

Simple algorithm for polarity detection

New sentiment measure

New score is (Nice +1) + (Not bad +1) + (Very uninspired -2) + (Too expensive -2) + (Sorely disappointed -2) + (Avoid -1) = -5

Easier and harder Problems

- **Tweets** from Twitter are probably the easiest, short and thus usually straight to the point

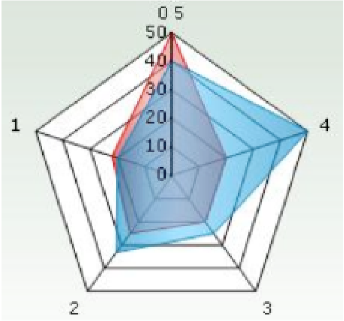
- **Reviews** are next, entities are given (almost) and there is little noise

- **Discussions**, **comments**, and **blogs** are hard.
    - Multiple entities, comparisons, noisy, sarcasm, etc
    - Determining sentiments seems to be easier.
    - Extracting entities and aspects is harder.
    - Combining them is even harder.

A structure (i.e., formalization) of the problem is needed for harder tasks.

Example of Hard Problem

"“I bought an iPhone a few days ago. It is such a nice phone. The touch screen is really cool. The voice quality is clear too. It is much better than my old Blackberry, which was a terrible phone and so difficult to type with its tiny keys. However, my mother was mad with me as I did not tell her before I bought the phone. She also thought the phone was too expensive, ...”



Feature Based Summary of iPhone:

Feature1: Touch screen

Positive: 212

The touch screen was really cool.

The touch screen was so easy to use and can do amazing things.

...

Negative: 6

The screen is easily scratched.

I have a lot of difficulty in removing finger marks from the touch screen.

... Feature2: voice quality

A structure (i.e., formalization) of the problem is needed for harder tasks.

Formalization

1. **Opinion definition. What is an opinion?**
   - Can we provide a structured definition?
   - If we cannot structure a problem, we probably do not understand the problem.
2. **Opinion summarization**
   - Opinions are subjective. An opinion from a single person is often not sufficient for action.
   - We need opinions from many people, and thus opinion summarization.

Formalization

User Id: Abc123, Date: 5-1-2008
Review:"*I bought an **iPhone** a few days ago. It is such a **nice phone**. The **touch screen** is **really cool**. The **voice quality** is **clear** too. It is much **better** than my **old Blackberry**, which was a **terrible phone** and so **difficult to type** with its **tiny keys**. However**, my mother** was **mad** with me as I did not tell her before I bought the **phone**. She also thought the **phone** was **too expensive**, ...*"

Different levels of granularity: one can look at this review/blog at the

1. document level, i.e., is this review + or – ?
2. sentence level, i.e., is each sentence + or - ?
3. entity and feature/aspect level

Formalization

User Id: **Abc123**, **Date:** **5-1-2008**
Review:“*I bought an **iPhone** a **few days ago**. It is such a **nice phone**. The **touch screen** is **really cool**. The **voice quality** is **clear** too. **It** is much **better** than **my old Blackberry**, which was a **terrible phone** and so **difficult to type** with its **tiny keys**. However, **my mother** was **mad** with me as I did not tell her before I bought the **phone**. **She** also thought the **phone** was **too expensive**, ...*”

What do we see?

- **Opinion targets**: entities and their features/aspects
- **Sentiments**: positive and negative
- **Opinion holders**: persons who hold the opinions
- **Time**: when opinions are expressed

Formalization

Opinion: an opinion has the following basic components

$$(g_j, so_{ijl}, h_i, t_l),$$

where

- $\mathbf{g_j}$ is a target (e.g., iPhone)
- $\mathbf{so_{ijl}}$ is the sentiment value of the opinion from opinion holder $\mathbf{h}_i$ on target $\mathbf{g}_j$ at time $\mathbf{t}_l$.
- $\mathbf{so}_{ijl}$ is positive, negative or neutral, or a rating score
- $\mathbf{h}_i$ is an opinion holder.
- $\mathbf{t}_l$ is the time when the opinion is expressed.

Formalization

In some cases, opinion target is a single entity or topic.

"I love iPhone"

But in many other cases, it is more complex.

"I bought an iPhone a few days ago. It is such a nice phone. The touch screen is really cool."

Opinion target of the 3rd sentence is not just touch screen, but the "touch screen of iPhone". We decompose the opinion target in ***entity*** and ***aspects***.

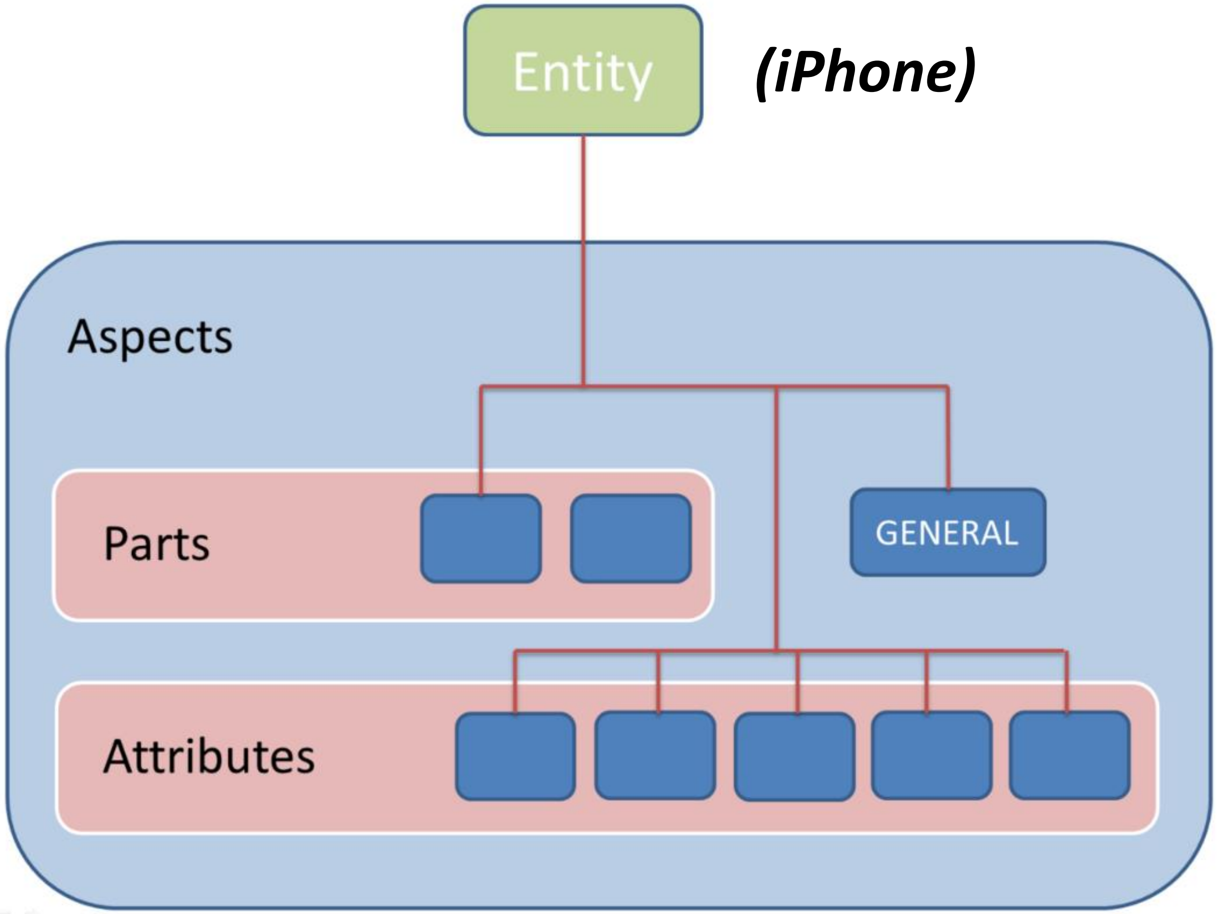
Formalization

Definition (**entity**): An *entity e* is a product, person, event, organization, or topic.
e is represented as

- a hierarchy of ***components***, sub-components, and so on (e.g., touch screen)
- each node represents a component and is associated with a set of ***attributes*** of the component (e.g., battery life, weight, size)

An opinion can be expressed on any node or attribute of the node.
For simplicity, we use the term ***aspects*** (features) to represent both components (or parts) and attributes.

Formalization



The special aspect **"GENERAL"** is used when the sentiment is expressed for the whole entity.

In this case, either the entity ***e*** and the aspect ***a*** represent the opinion target.

Formalization

Opinion definition: an opinion is a quintuple

$$(e_j, a_{jk}, so_{ijkl}, h_i, t_l),$$

where

- $\mathbf{e_j}$ is a target entity;
- $\mathbf{a_{jk}}$ is an aspect/feature of the entity $\mathbf{e_j}$ $\mathbf{so_{ijkl}}$ is the sentiment value of the opinion from opinion holder $\mathbf{h}_i$ on aspect $\mathbf{a}_{jk}$ of entity $\mathbf{e}_j$ at time $\mathbf{t}_l$. $\mathbf{so}_{ijkl}$ is positive, negative or neutral, or a rating value;
- $\mathbf{h}_i$ is an opinion holder;
- $\mathbf{t}_l$ is the time when the opinion is expressed.

Formalization

User Id: Abc123, Date: 5-1-2008
Review:“*I bought an **iPhone** a few days ago. It is such a **nice phone**. The **touch screen** is **really cool**. The **voice qualit**y is **clear** too. It is much **better** than my **old Blackberry**, which was a **terrible phone** and so **difficult to type** with its **tiny keys**. However**, my mother** was **mad** with me as I did not tell her before I bought the **phone**. She also thought the **phone** was **too expensive**, ...*”

In quintuples ***(entity, aspect, sentiment, holder, time)***

– (iPhone, GENERAL, +, Abc123, 5-1-2008)
– (iPhone, touch_screen, +, Abc123, 5-1-2008)
– (iPhone, GENERAL, -, my mother, 5-1-2008)
– ...

The Quintuple is hard to resolve

$$(e_j, a_{jk}, so_{ijkl}, h_i, t_l),$$

- $\mathbf{e_j}$ → Named Entity Extraction
- $\mathbf{a_{jk}}$ → Information Extraction
- $\mathbf{so_{ijkl}}$ → Sentiment Analysis
- $\mathbf{h}_i$ → Information Extraction
- $\mathbf{t}_l$ → Time Extraction

The most of these problems are yet unsolved in computer science *(see Bing Liu)*

The Quintuple is hard to resolve

"As much use as a trapdoor on a lifeboat" - negative but not obvious to the machine.

"The canon camera is better than the Fisher Price one" - comparisons are hard to classify.

"imo the ice cream is luuurrrrrrvely" - slang and the way we communicate in general needs to be processed.

The Quintuple is hard to resolve

Goal: Given an opinionated document

- Discover all quintuples
- Or, solve some simpler forms of the problem
    - E.g., sentiment classification at the document or sentence level.

- With the quintuples,
  – Unstructured Text → Structured Data
  – Traditional data and visualization tools can be used to slice and visualize the results.
  – Enable qualitative and quantitative analysis.

Use of Machine Learning

Intro to NLTK and scikit-learn

Unsupervised Learning
Inferring a function to describe hidden structure from unlabeled data. The examples given to the learner are unlabeled.

Supervised Learning
Inferring a function to describe hidden structure from labeled data. The training data consist of a set of training examples (input object and desired output value). SVM, Naive Bayesian Classifiers, etc.

Use of Machine Learning
(step-by-step example)

First, we construct a list of documents, labeled with the appropriate categories. For this example, choose the Movie Reviews Corpus (available in NLTK), which categorizes each review as positive or negative.

```
>>> from nltk.corpus import movie_reviews
>>> documents = [(list(movie_reviews.words(fileid)), category)
...              for category in movie_reviews.categories()
...              for fileid in movie_reviews.fileids(category)]
>>> random.shuffle(documents)
```

Use of Machine Learning

Next, we define a feature extractor for documents. We can define a feature for each word, indicating whether the document contains that word.

To limit the number of, we consider a list of the 2000 most frequent words in the overall corpus and define a feature extractor that simply checks whether each of these words is present in a given document (0/1).

```
all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

Use of Machine Learning

Now that we've defined our feature extractor, we can use it to train a classifier to label new movie reviews.

To check how reliable the resulting classifier is, we compute its accuracy on the test set. And once again, we can use **show_most_informative_features()** to find out which features the classifier found to be most informative.

```
featuresets = [(document_features(d), c) for (d,c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

Use of Machine Learning

Apparently in this corpus, a review that mentions "Seagal" is almost 8 times more likely to be negative than positive, while a review that mentions "Damon" is about 6 times more likely to be positive.

```
>>> print(nltk.classify.accuracy(classifier, test_set))
0.81
>>> classifier.show_most_informative_features(5)
Most Informative Features
   contains(outstanding) = True              pos : neg    =     11.1 : 1.0
        contains(seagal) = True              neg : pos    =      7.7 : 1.0
   contains(wonderfully) = True              pos : neg    =      6.8 : 1.0
         contains(damon) = True              pos : neg    =      5.9 : 1.0
        contains(wasted) = True              neg : pos    =      5.8 : 1.0
```

Use of Machine Learning

Bag of Words (BoW)

A very common feature extraction procedures for sentences and documents is the bag-of-words approach (BOW). In this approach, we look at the histogram of the words within the text, i.e. considering each word count as a feature.

Page 69 - Goldberg, Yoav. "Neural network methods for natural language processing." Synthesis Lectures on Human Language Technologies 10.1 (2017): 1-309.

Use of Machine Learning

Bag of Words (BoW)

Once a vocabulary has been chosen, the occurrence of words in example documents needs to be scored.

A very simple approach is a binary scoring of the presence or absence of words.

Some additional simple scoring methods include:

- **Counts**. Count the number of times each word appears in a document.
- **Frequencies**. Calculate the frequency that each word appears in a document out of all the words in the document.

Use of Machine Learning

`sklearn.feature_extraction.text`.CountVectorizer

class `sklearn.feature_extraction.text.` **`CountVectorizer`** (*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>*) [source]

Convert a collection of text documents to a matrix of token counts

This implementation produces a sparse representation of the counts using scipy.sparse.csr_matrix.

If you do not provide an a-priori dictionary and you do not use an analyzer that does some kind of feature selection then the number of features will be equal to the vocabulary size found by analyzing the data.

[Documentation Link]

Use of Machine Learning

Example

```
from sklearn.feature_extraction.text import CountVectorizer

corpus = [
'All my cats in a row',
'When my cat sits down, she looks like a Furby toy!',
'The cat from outer space',
'Sunshine loves to sit like this for some reason.'
]

vectorizer = CountVectorizer()
print( vectorizer.fit_transform(corpus))
print( vectorizer.vocabulary_ )
```

Use of Machine Learning

TF-IDF (Term Frequency - Inverse Document Frequency): is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

Term frequency: the number of times a term occurs in a document.
Inverse Document Frequency: is a measure of how much information the word provides, that is, whether the term is common or rare across all documents.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

Use of Machine Learning

sklearn.feature_extraction.text.TfidfVectorizer

class `sklearn.feature_extraction.text.` **`TfidfVectorizer`** (*input=’content’, encoding=’utf-8’, decode_error=’strict’, strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer=’word’, stop_words=None, token_pattern=’(?u)\b\w\w+\b’, ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class ‘numpy.int64’>, norm=’l2’, use_idf=True, smooth_idf=True, sublinear_tf=False*) [source]

Convert a collection of raw documents to a matrix of TF-IDF features.

Equivalent to CountVectorizer followed by TfidfTransformer.

[Documentation Link]

Use of Machine Learning

Another example (todo):

- Download movie reviews from the following URL (opinions about “The Da Vinci Code” book and “Harry Potter” movie labeled as positive or negative)
  http://www.dmi.unict.it/ortis/ PhDCourseSentiment/davincireviews.txt
- Vectorize the text using tf-idf (Term Frequency – Inverse Document Frequency)

  skipping stopwords
- Split dataset in X_train, X_test, Y_train, Y_test (Y variable is 0 or 1)
- Train a Naive Bayes classifier with X_train and Y_train
- Test model with Y_test, X_test

Lexical Resources

WordNet is a lexical database for the English language. It groups English words into sets of synonyms called synsets, provides short definitions and usage examples, and records a number of relations among these synonym sets or their members. WordNet can thus be seen as a combination of dictionary and thesaurus.

SentiWordNet is a lexical resource for opinion mining. SentiWordNet assigns to each synset of WordNet three sentiment scores: positivity, negativity, objectivity.

Natural Language Processing

From text to sentiments

The extraction of sentiment is based on the detection (counting) of words with certain positive or negative polarity by means of specific lexicons and linguistic resources.

SentiWordNet

synsets	positive	negative	objective
good#1	0.75	0	0.25
divine#1	0.875	0	0.125
solid#1	0.875	0	0.125
superb#2	0.875	0	0.125

Lexical Resources

SentiStrength

Automatic sentiment analysis of up to 16,000 social web texts per second with up to human level accuracy for English - other languages available or easily added.

SentiStrength estimates the *strength* of positive and negative sentiment in *short texts*, even for informal language. It has human-level accuracy for short social web texts in English, except political texts. SentiStrength reports *two* sentiment strengths:

-1 (not negative) to -5 (extremely negative)

1 (not positive) to 5 (extremely positive)

Why does it use two scores? Because research from psychology has revealed that we process positive and negative sentiment in parallel - hence mixed emotions.

Resources

- Natural Language Toolkit: http://www.nltk.org
- WordNet: https://wordnet.princeton.edu
- SentiWordNet: http://sentiwordnet.isti.cnr.it
- SentiStrenght: http://sentistrength.wlv.ac.uk/