FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Grado en Inteligencia Artificial

Review – Text Mining

Prof. Dante Conti

Prof. Sergi Ramírez

Investigación: por ejemplo, el descubrimiento de conocimientos, la atención médica y sanitaria: en el pasado, a un investigador humano le lleva mucho tiempo analizar y obtener información relevante. En algunos casos, esta información ni siquiera era accesible. La minería de textos permite a los investigadores encontrar más información y de forma más rápida y eficiente.

Negocios: por ejemplo, las grandes empresas utilizan la minería de textos para ayudar en la toma de decisiones y responder rápidamente a las consultas de los clientes en procesos tales como la gestión de riesgos o el filtrado de currículos

Seguridad: En anti-terrorismo, el análisis de los blogs y otras fuentes de texto en línea se utiliza para prevenir delitos en Internet y luchar contra el fraude.

Diariamente, La minería de texto es usada por los sitios web de correo electrónico para crear métodos de filtrado más confiables y efectivos, **para el filtrado de spam, análisis de datos de medios sociales, etc. También para identificar las relaciones entre los usuarios y ciertos productos o para determinar las opiniones de los usuarios sobre temas particulares**

Fases de la minería de textos

La minería de textos es el proceso encargado del descubrimiento de información que no existía explícitamente en ningún texto de la colección, pero que surge de relacionar el contenido de varios de ellos. Para ello, la minería de textos comprende tres actividades fundamentales:

1. **Recuperación de la información**: Consiste en seleccionar los textos pertinentes
2. **Extracción de la información** incluida en esos textos mediante el procesamiento del lenguaje natural: Hechos, acontecimientos, datos clave, relaciones entre ellos, etc.
3. **Minería de datos** para encontrar asociaciones entre los datos clave previamente extraídos de entre los textos.

Estas actividades las dividimos dentro de tres etapas fundamentales:

- **Etapa de pre-procesamiento**: En esta etapa los textos se transforman en algún tipo de representación estructurada o semi-estructurada que facilite su posterior análisis. Es decir, el primer paso dentro de la minería de texto sería definir el conjunto (corpus) de documentos. Estos documentos deben ser **representativo** y seleccionarse **aleatoriamente** o mediante algún método de muestreo probabilístico. Se debe evitar en esta etapa la duplicación de documentos dentro del corpus.

Una vez que hemos seleccionado los documentos a analizar, el siguiente paso será convertirlos a un formato analizable, para poder crear una representación estructurada o semi-estructurada de los mismos.

Con el corpus seleccionado y estructurado, debemos reconocer los **tokens** (unidades gramaticales más pequeñas), lo que implica representar el texto como una lista de palabras mediante una representación vectorial.

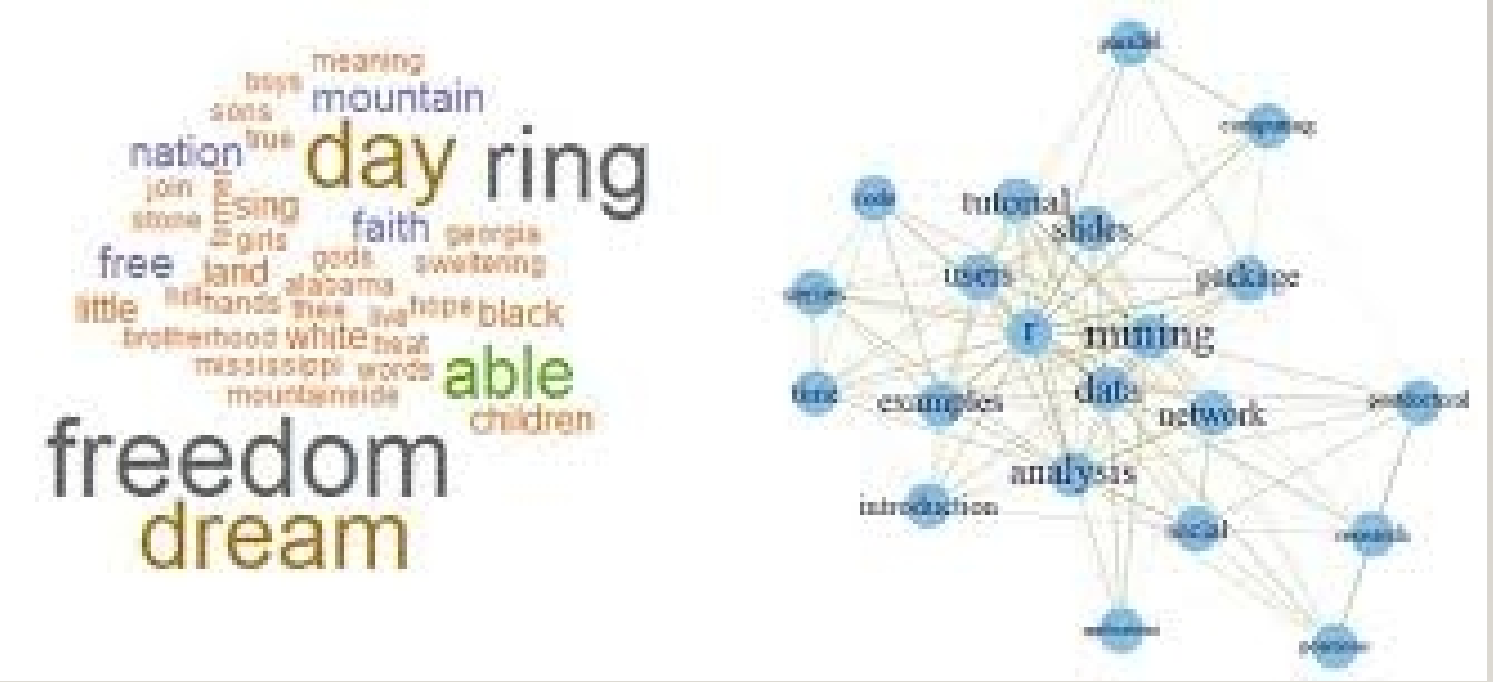
- **Etapa de descubrimiento**: En esta etapa las representaciones internas se analizan con el objetivo de descubrir en ellas algunos patrones interesantes o nueva información.

- **Etapa de visualización**: Es la etapa en la que los usuarios pueden observar y explorar los resultados.

Aplicaciones del Text Mining

En pocas palabras, el text mining se podrá aplicar para:

- La extracción de información
- El análisis de sentimientos o minería de opiniones
- La clasificación documental
- La elaboración de resúmenes
- La extracción de conocimiento



1. Librerías necesarias:

```
library(tm)
library(Snowball)
library(slam)
library(Matrix) # para trabajar con matrices
```

2. Carga de datos en R

Partiendo de un fichero .csv (comma separated values), donde en una de las columnas se encuentran los textos (como por ejemplo el contenido de un tuit), añadimos los nombres de las columnas y creamos lo que para la librería tm (text mining) sería un Corpus:

```
txtrepo = read.csv("datos.csv") cols <- readLines("colnames.txt",encoding="UTF-8")
colnames(txtrepo) <- cols
```

Como un preproceso inicial, eliminamos las columnas vacías (como por ejemplo tuits sin texto en la columna descripción):

```
txtdataset <- subset(txtrepo, descripcion!="\\N")
```

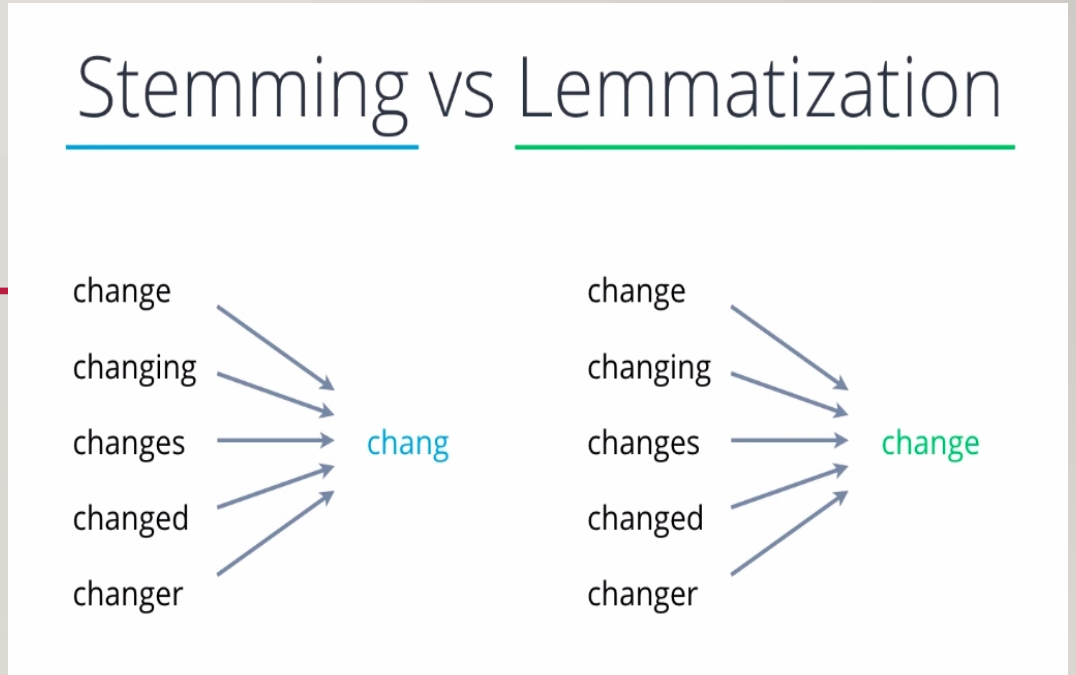
Creamos el corpus a partir del vector (o columna) descripción:

```
micorpus <- Corpus(VectorSource(txtdataset$descripcion), readerControl = list(reader = readPlain, language = "es", load = TRUE))
```

3. Preprocesado de datos

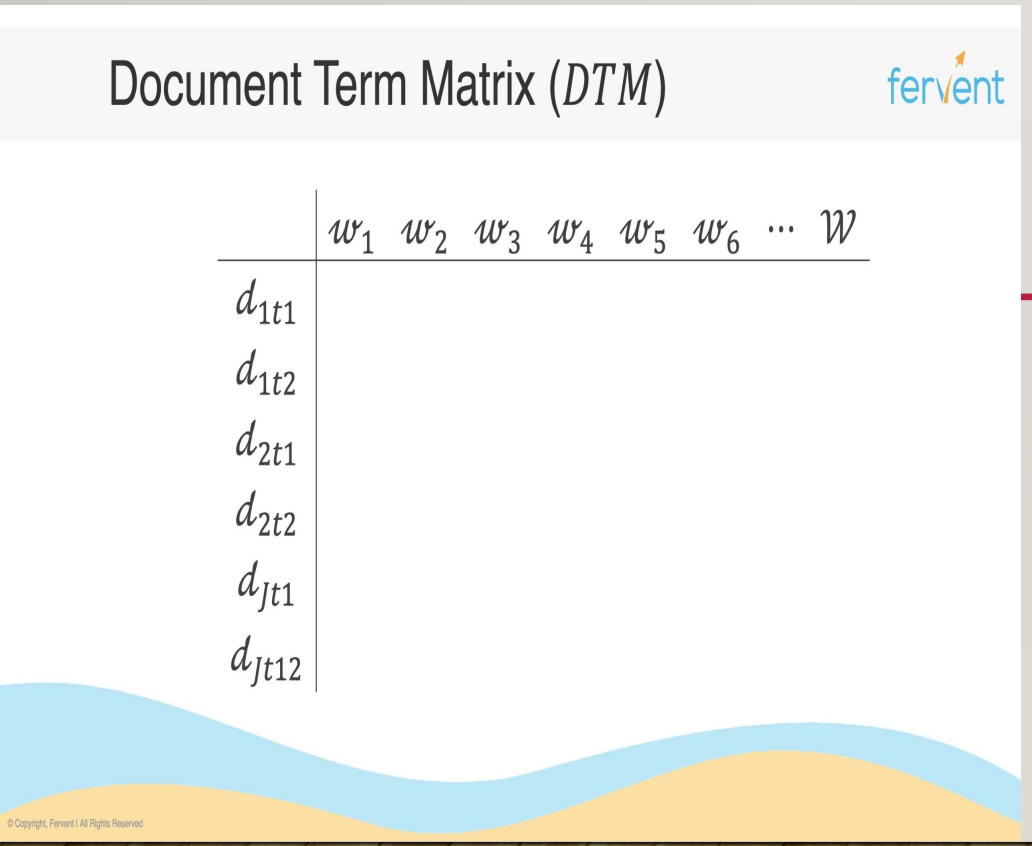
Tenemos un vector con el texto en crudo a clasificar, pero antes de nada es necesario preprocesarlo para eliminar e igualar los términos. Así, quitamos los números, los signos de puntuación, palabras de parada (artículos, pronombres,…), lo ponemos todo en minúscula, quitamos espacios en blanco sobrante, y aplicamos stemming (reducir las palabras a su raíz, como de “clasificar” a “clasific”).

```
micorpus <- tm_map(micorpus, removeNumbers)
micorpus <- tm_map(micorpus, removePunctuation)
micorpus <- tm_map(micorpus, tolower)
micorpus <- tm_map(micorpus, removeWords, stopwords("spanish"))
micorpus <- tm_map(micorpus, stemDocument, language="spanish")
micorpus <- tm_map(micorpus, stripWhitespace)
```



Después creamos lo que se conoce como una matriz de términos del documento (**Document Term Matrix**). Se trata de una matriz de dimensión *mxn*, donde *m* sería el número de descripciones (o documentos, o tuits,…) a procesar, y *n* sería el número de términos existentes en esos documentos. Los valores de la matríz sería el número de veces que cada documento (fila) contiene el término (columna) dado.

```
dtm <- DocumentTermMatrix(micorpus)
dtm.data <- as.matrix(dtm) # Lo mismo pero en formato de dataframe
```



1. Binary weights
2. Term Frequency (TF)
3. Inverse Document Frequency (IDF)
4. Term Frequency-Inverse Document Frequency (TF-IDF)

1. Binary Weights

In the case of binary weights, the weights take the values- 0 or 1 where 1 reflects the presence and 0 reflects the absence of the term in a particular document.

For instance,

D1: Text mining is to find useful information from text.

D2: Useful information is mined from the text.

D3: Dark came.

	text	mining	is	to	find	useful	information	from	text	mined	dark	came
D1	1	1	1	1	1	1	1	1	1	0	0	0
D2	0	0	1	0	0	1	1	1	1	1	0	0
D3	0	0	0	0	0	0	0	0	0	0	1	1

2. Term Frequency

In the case of the term Frequency, the weights represent the frequency of the term in a specific document. The underlying assumption is that the higher the term frequency in a document, the more important it is for that document.

3. Inverse Document Frequency

In the case of IDF, the underlying idea is to assign higher weights to unusual terms, i.e., to terms that are not so common in the corpus. IDF is computed at the corpus level, and thus describes corpus as a whole, not individual documents. It is computed in the following way:

$$IDF(t)=1+\log(N/df(t))$$

N- number of documents in the corpus

Df(t)- number of documents with the term t

For instance, suppose there are 100 documents in the corpus and 10 documents contain the term text.

Then, $IDF(text)=1+\log(100/10)=1+1=2$

4. TF-IDF

In the case of TF-IDF, the underlying idea is to value those terms that are not so common in the corpus (relatively high IDF), but still have some reasonable level of frequency (relatively high TF). It is the most frequently used metric for computing term weights in a vector space model.

General formula for computing TF-IDF:

$$TF\text{-}IDF(t)=TF(t)*IDF(t)$$

One popular ‘instantiation’ of this formula:

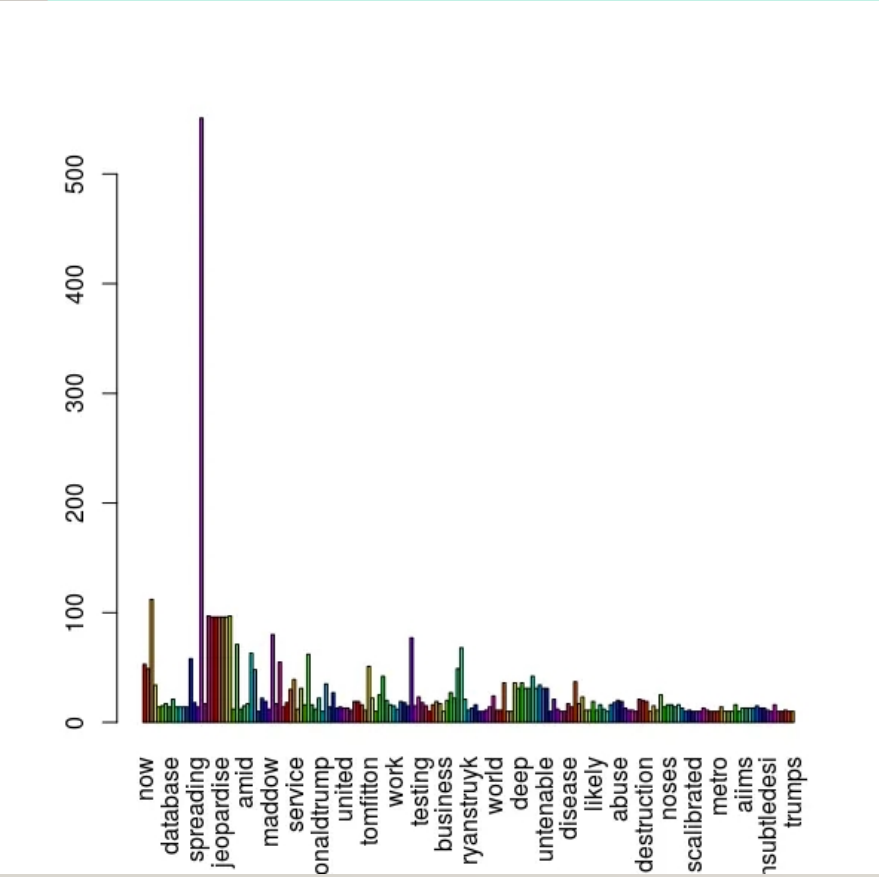
$$TF\text{-}IDF(t)= tf(t)*\log(N/df(t))$$

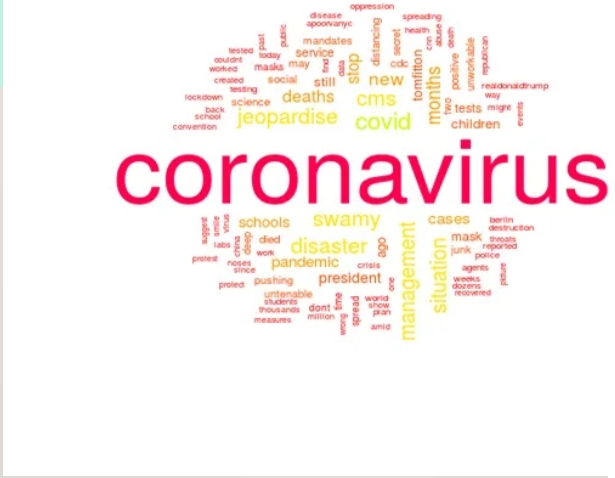
4. Análisis estadístico (Similitudes entre palabras – Similitudes entre Documentos

Finalmente, como he comentado al inicio, si el objetivo es únicamente agrupar los tuits más parecidos, un simple análisis estadístico comparando las distancias entre documentos resultaría suficiente. Por ejemplo, se trataría de crear una nueva matriz de distancias, siendo el resultado una matriz cuadrada de distancias (**Distance Matrix**) *nxn*, donde computamos la distancia de cada documento (filas) con el resto de documentos (columnas). Así, empezaría calculando la distancia (Euclídea, Manhattan, Binaria,…) entre el primer documento y él mismo (sindo su distancia 0 ya que son el mismo documento), poniendo su valor en la primera casilla de la matriz:

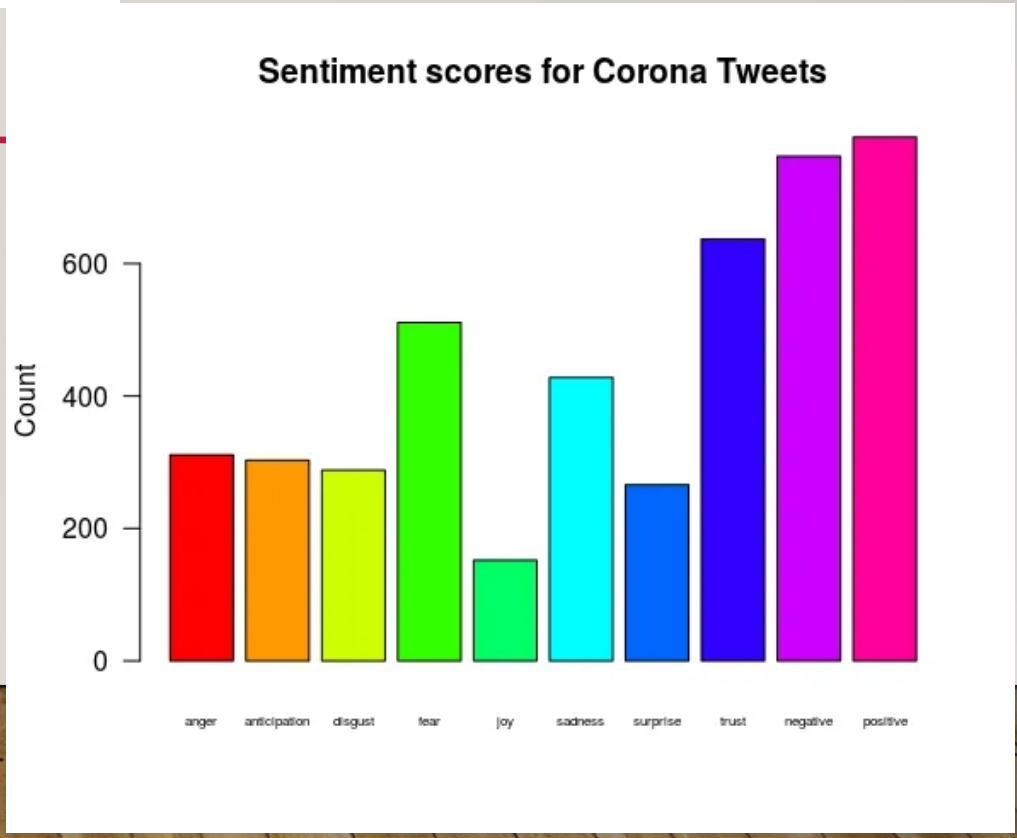
```
distancias <- as.matrix(dist(dtm.data, method = "binary", diag = TRUE, upper = FALSE, p = 2))
```

5. Análisis y Modelos





	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
1	0	0	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	2	0	3
4	0	0	1	0	0	1	0	0	1	2
5	0	1	0	1	0	0	0	1	1	1
6	1	0	1	1	0	1	1	1	1	1



MORE..

Rapid Automatic Keyword **Extraction**(**RAKE**) → uses a list of stopwords and phrase delimiters to detect the most relevant words or phrases in a piece of text.

Pointwise Mutual Information (PMI) → More likely pair of words

TextRank – is a graph-based ranking model for text processing which can be used in order to find the most relevant sentences in text and also to find keywords.

Word embeddings

t-distributed Stochastic Neighbor Embedding

WHAT IS SENTIMENT ANALYSIS? - OPTIONAL TOPIC

Sentiment analysis is the process of detecting positive or negative sentiment in text. It's often used by businesses to detect sentiment in social data, gauge brand reputation, and understand customers.

Sentiment analysis focuses on the polarity of a text (*positive, negative, neutral*) but it also goes beyond polarity to detect specific feelings and emotions (*angry, happy, sad*, etc), urgency (*urgent, not urgent*) and even intentions (*interested v. not interested*).

TYPES OF SENTIMENT ANALYSIS

Graded Sentiment Analysis

If polarity precision is important to your business, you might consider expanding your polarity to include different levels of positive and negative categories.

Emotion detection

Emotion detection sentiment analysis allows you to go beyond polarity to detect emotions, like happiness, frustration, anger, and sadness.

TYPES OF SENTIMENT ANALYSIS

Aspect-based Sentiment Analysis

Usually, when analyzing sentiments of texts, you'll want to know which aspects or features people are mentioning in a positive, neutral, or negative way. For example, in this product review: *"The battery life of this camera is too short"*, an aspect-based classifier would be able to determine that the sentence expresses a negative opinion about the battery life of the product in question.

Multilingual sentiment analysis

CHALLENGES OF SENTIMENT ANALYSIS

To understand the goal and challenges of sentiment analysis, here are some examples:

Basic examples of sentiment analysis data

- Netflix has the best selection of films
- Hulu has a great UI
- I dislike like the new crime series
- I hate waiting for the next series to come out

More challenging examples of sentiment analysis

- I *do not dislike* horror movies. (phrase with negation)
- Disliking horror movies is *not* uncommon. (negation, inverted word order)
- *Sometimes* I really hate the show. (adverbial modifies the sentiment)
- I love having to wait two months for the next series to come out! (sarcasm)
- The final episode was surprising with a *terrible* twist at the end (negative term used in a positive way)
- The film was easy to watch but I would not recommend it to my friends. (difficult to categorize)
- I *LOL'd* at the end of the cake scene (often hard to understand new terms)

METHODS

A subjectivity lexicon is a predefined list of words associated with emotional context such as positive/negative (BING LEXICON, AFINN, NRC)

Rule-based Approaches → rule-based system uses a set of human-crafted rules to help identify subjectivity, polarity, or the subject of an opinion.
Stemming, *tokenization*, *part-of-speech, tagging* and *parsing*.
Lexicons (i.e. lists of words and expressions).

Here's a basic example of how a rule-based system works:

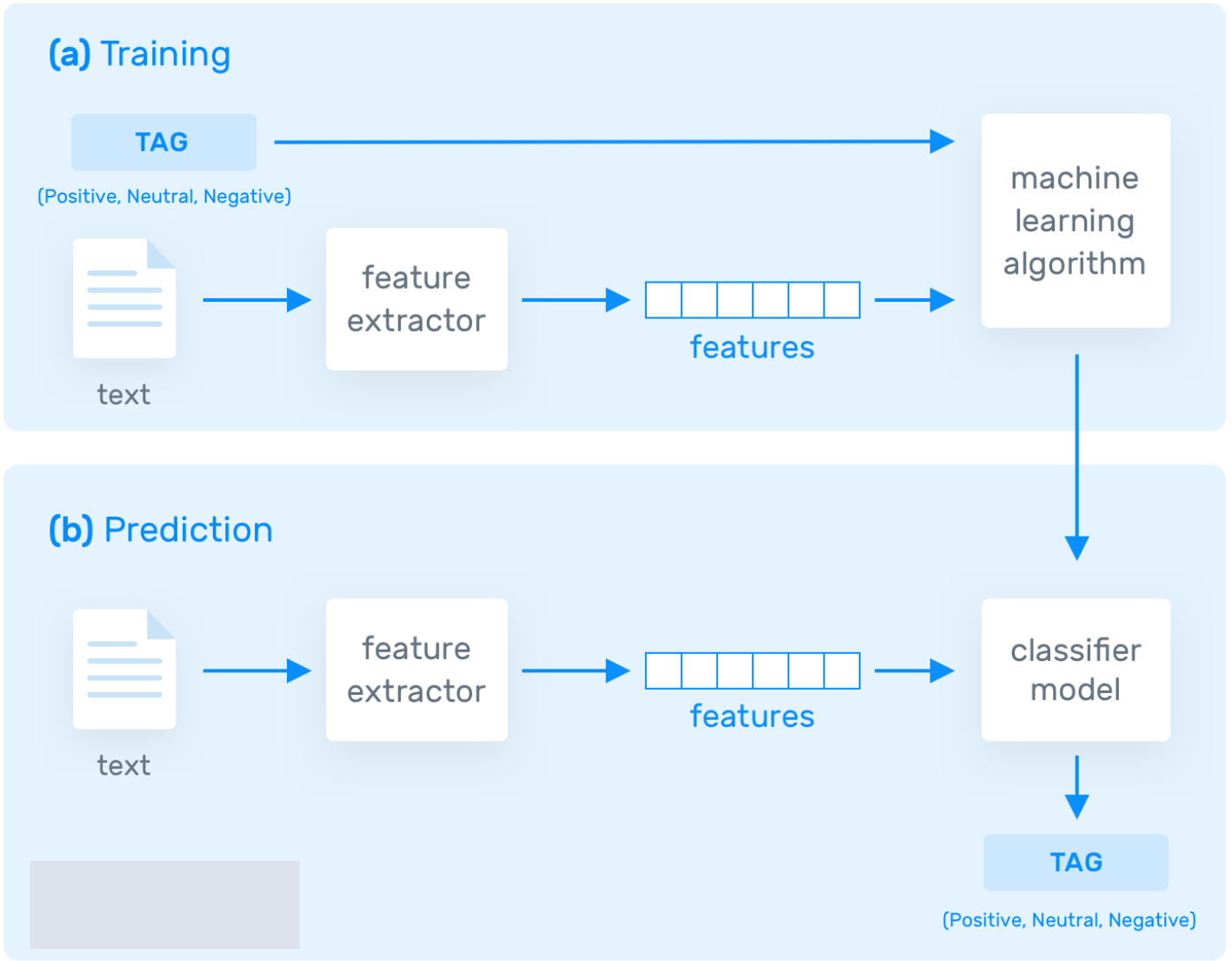
1. Defines two lists of polarized words (e.g. negative words such as *bad*, *worst*, *ugly*, etc and positive words such as *good*, *best*, *beautiful*, etc).
2. Counts the number of positive and negative words that appear in a given text.
3. If the number of positive word appearances is greater than the number of negative word appearances, the system returns a positive sentiment, and vice versa. If the numbers are even, the system will return a neutral sentiment.

Rule-based systems are very naive since they don't take into account how words are combined in a sequence. Of course, more advanced processing techniques can be used, and new rules added to support new expressions and vocabulary. However, adding new rules may affect previous results, and the whole system can get very complex. Since rule-based systems often require fine-tuning and maintenance, they'll also need regular investments.

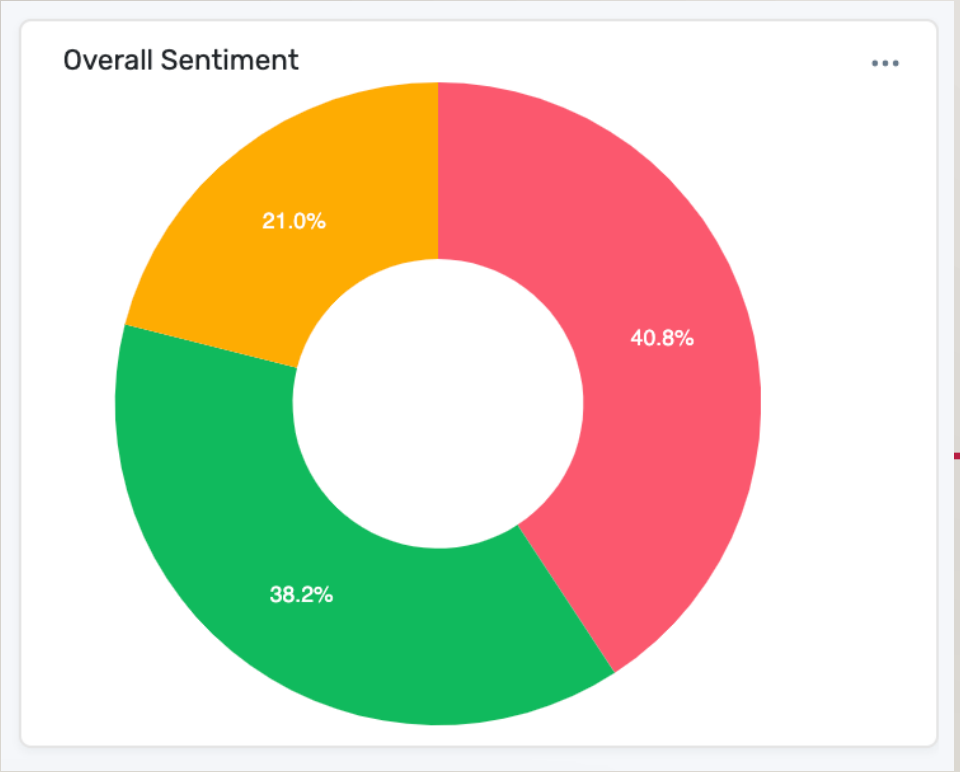
METHODS

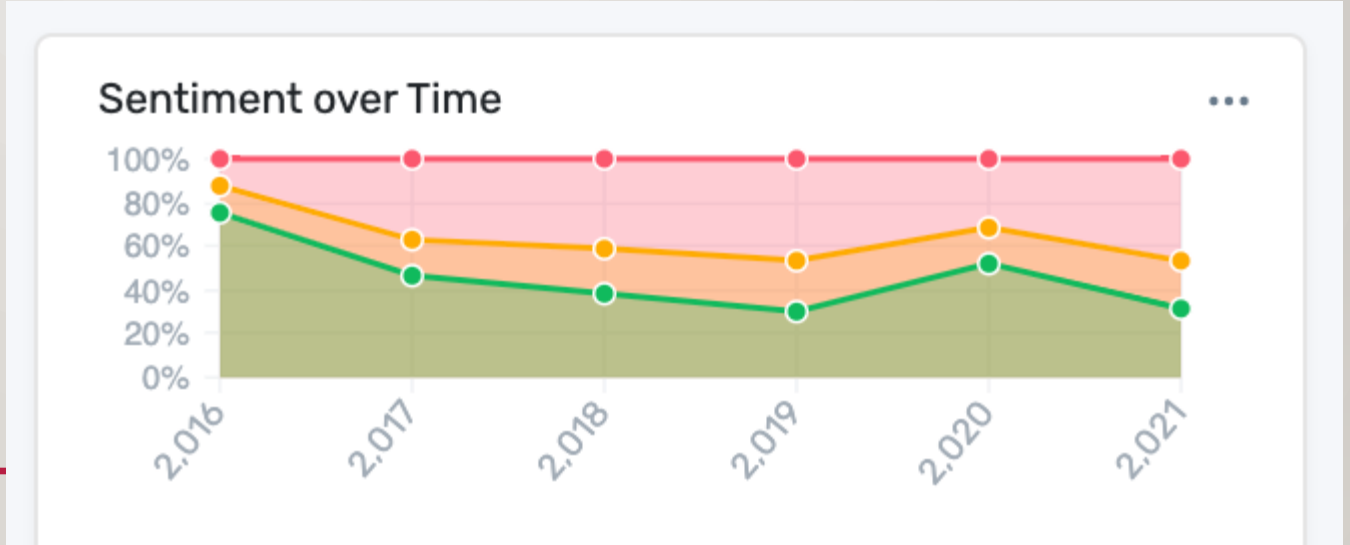
Automatic Approaches

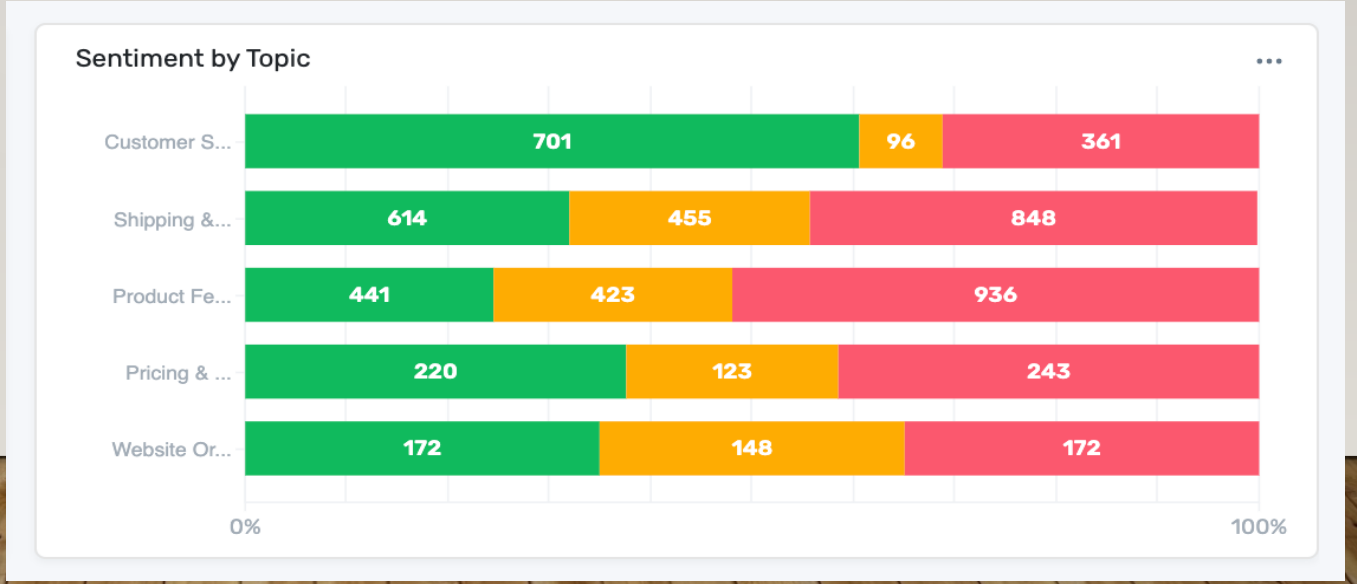
Automatic methods, contrary to rule-based systems, don't rely on manually crafted rules, but on machine learning techniques. A sentiment analysis task is usually modeled as a classification problem, whereby a classifier is fed a text and returns a category, e.g. positive, negative, or neutral.



ORGANIZING RESULTS







ORGANIZING RESULTS