# Introducción a la minería de textos y procesamiento de lenguaje natural para ciencias sociales

## Clase 2. Análisis de Sentimiento + Vectorización de texto

factor~data
IDAES_UNSAM

# Análisis de Sentimiento

factor~data
IDAES_UNSAM

# ¿Qué es?

- Área dentro del PLN que busca conocer el “sentimiento” expresado en un texto,
- Usa como base las palabras y expresiones contenidas en el mismo.
- Usos comunes: medición de la opinión de consumidores sobre artículos, bienes y/o servicios
- Otros usos

# ¿Qué es?

**Alejandro Fantino**
@FantinoFantino

Qué gran jugador es Messi. El partido de ayer fue genial. Y el segundo gol, glorioso.

12:00 PM · Jun 1, 2021

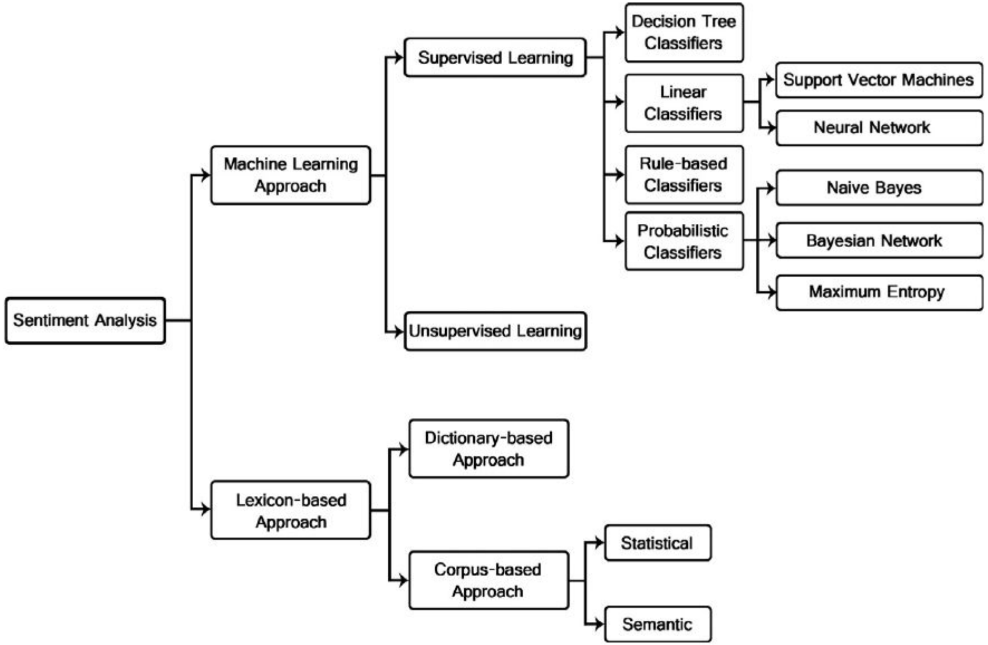
**Colorado Lieberman**
@colorado

Messi es un desastre. Y la selección, triste.

12:00 PM · Jun 1, 2021

# ¿Qué es?

- Tres grandes enfoques:
    - Basados en léxico -lexicon based-
    - Basados en Aprendizaje Automático -automatic learning-
    - Enfoques híbridos -hybrid-

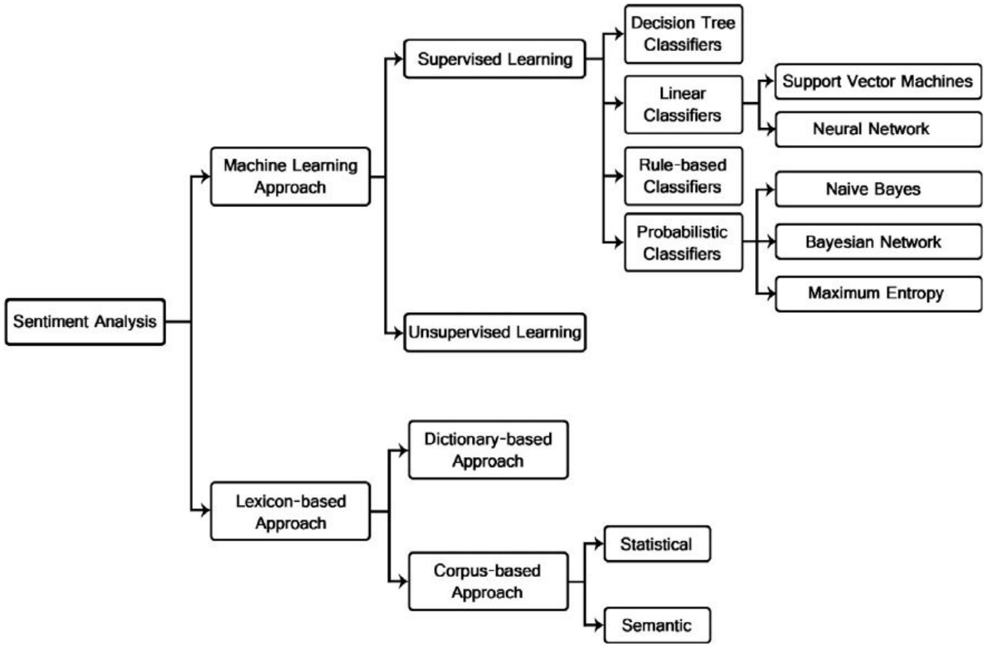



factor~data
IDAES_UNSAM

# Métodos basados en léxicos

- Se usan listados de palabras o expresiones con connotación conocida para clasificar un texto dado.

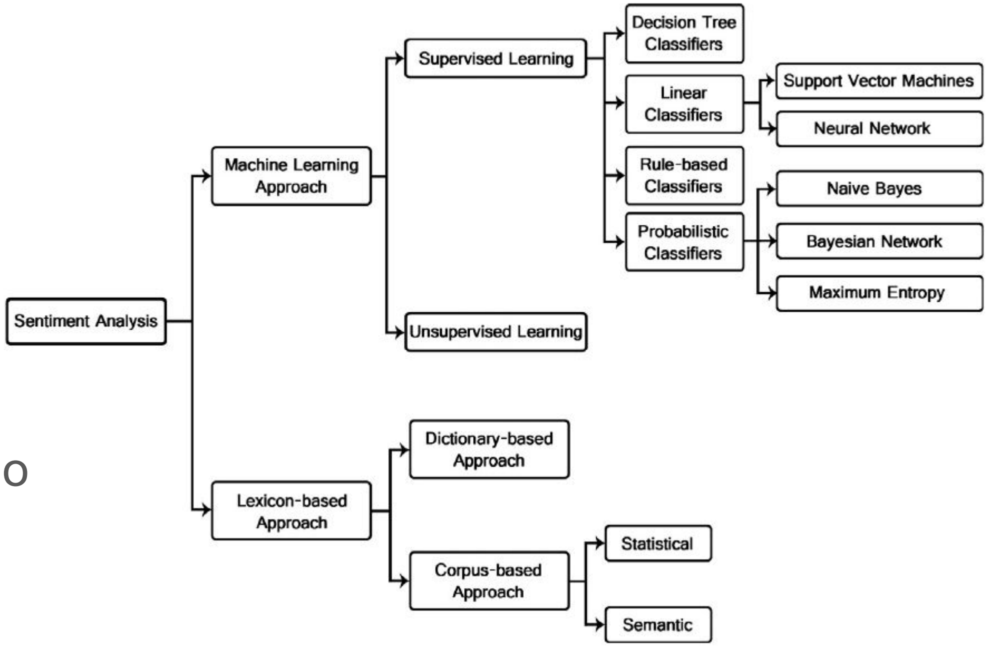
- “A mano” para cada corpus
- Lexicones generados por otras personas o procesos
- Enfoque basado en corpus




factor~data
IDAES_UNSAM

# Métodos basados en Machine Learning

- Técnicas de ML para clasificar textos.
- Tenemos un conjunto de texto preclasificado (+, -, etc.)
- Se entrena un modelo que relacionar e identifique features relevantes a las clases
- Se utiliza el modelo ya entrenado para predecir el sentimiento de nuevos textos



**Sentiment Analysis** | **Information** | Live Demo | Sentiment Treebank | Help the Model | Source Code

### Deeply Moving: Deep Learning for Sentiment Analysis

This website provides a live demo for predicting the sentiment of movie reviews. Most sentiment prediction systems work just by looking at words in isolation, giving positive points for positive words and negative points for negative words and then summing up these points. That way, the order of words is ignored and important information is lost. In constrast, our new deep learning model actually builds up a representation of whole sentences based on the sentence structure. It computes the sentiment based on how words compose the meaning of longer phrases. This way, the model is not as easily fooled as previous models. For example, our model learned that *funny* and *witty* are positive but the following sentence is still negative overall:

*This movie was actually neither that funny, nor super witty.*

The underlying technology of this demo is based on a new type of *Recursive Neural Network* that builds on top of grammatical structures. You can also browse the Stanford Sentiment Treebank, the dataset on which this model was trained. The model and dataset are described in an upcoming EMNLP paper. Of course, no model is perfect. You can help the model learn even more by labeling sentences we think would help the model or those you try in the live demo.

Paper Title and Abstract

**Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank**
Semantic word spaces have been very useful but cannot express the meaning of longer phrases in a principled way. Further progress towards understanding compositionality in tasks such as sentiment detection requires richer supervised training and evaluation resources and more powerful models of composition. To remedy this, we introduce a Sentiment Treebank. It includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences and presents new challenges for sentiment compositionality. To address them, we introduce the Recursive Neural Tensor Network. When trained on the new treebank, this model outperforms all previous methods on several metrics. It pushes the state of the art in single sentence positive/negative classification from 80% up to 85.4%. The accuracy of predicting fine-grained sentiment labels for all phrases reaches 80.7%, an improvement of 9.7% over bag of features baselines. Lastly, it is the only model that can accurately capture the effect of contrastive conjunctions as well as negation and its scope at various tree levels for both positive and negative phrases.

**Paper:** Download pdf

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng and Christopher Potts

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

*Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*

**Dataset Downloads:**

Main zip file with readme (6mb)
Dataset raw counts (5mb)
Train,Dev,Test Splits in PTB Tree Format

**Code:** Download Page

**Press:** Stanford Press Release

Dataset visualization and web design by Jason Chuang. Live demo by Jean Wu, Richard Socher, Rukmani Ravisundaram and Tayyab Tariq.

This webpage requires one of the following web browsers:

Tweet | Like 26

README.md

# pysentimiento: A Python toolkit for Sentiment Analysis and Social NLP tasks

run_tests passing

A Transformer-based library for SocialNLP classification tasks.

Currently supports:

- Sentiment Analysis (Spanish, English)
- Emotion Analysis (Spanish, English)

Just do `pip install pysentimiento` and start using it:

Open in Colab

```
from pysentimiento import SentimentAnalyzer
analyzer = SentimentAnalyzer(lang="es")

analyzer.predict("Qué gran jugador es Messi")
# returns SentimentOutput(output=POS, probas={POS: 0.998, NEG: 0.002, NEU: 0.000})
analyzer.predict("Esto es pésimo")
# returns SentimentOutput(output=NEG, probas={NEG: 0.999, POS: 0.001, NEU: 0.000})
analyzer.predict("Qué es esto?")
# returns SentimentOutput(output=NEU, probas={NEU: 0.993, NEG: 0.005, POS: 0.002})

analyzer.predict("jejeje no te creo mucho")
# SentimentOutput(output=NEG, probas={NEG: 0.587, NEU: 0.408, POS: 0.005})
"""
Emotion Analysis in English
"""

emotion_analyzer = EmotionAnalyzer(lang="en")

emotion_analyzer.predict("yayyy")
# returns EmotionOutput(output=joy, probas={joy: 0.723, others: 0.198, surprise: 0.038, disgust: 0.01
emotion_analyzer.predict("fuck off")
# returns EmotionOutput(output=anger, probas={anger: 0.798, surprise: 0.055, fear: 0.040, disgust: 0.
```

Also, you might use pretrained models directly with `transformers` library.

# Problemas

- Castellano… siempre las cosas andan mal.
- ¿Cómo preprocesamos?
- ¿Cómo construimos lexicones?
- ¿Cómo generamos una base de datos taggeada?

## Sentiment Analysis

| Information | Live Demo | Sentiment Treebank | Help the Model | Source Code

**Deeply Moving: Deep Learning for Sentiment Analysis**

This website provides a live demo for predicting the sentiment of movie reviews. Most sentiment prediction systems work just by looking at words in isolation, giving positive points for positive words and negative points for negative words and then summing up these points. That way, the order of words is ignored and important information is lost. In constrast, our new deep learning model actually builds up a representation of whole sentences based on the sentence structure. It computes the sentiment based on how words compose the meaning of longer phrases. This way, the model is not as easily fooled as previous models. For example, our model learned that *funny* and *witty* are positive but the following sentence is still negative overall:

*This movie was actually neither that funny, nor super witty.*

The underlying technology of this demo is based on a new type of *Recursive Neural Network* that builds on top of grammatical structures. You can also browse the Stanford Sentiment Treebank, the dataset on which this model was trained. The model and dataset are described in an upcoming EMNLP paper. Of course, no model is perfect. You can help the model learn even more by labeling sentences we think would help the model or those you try in the live demo.

Paper Title and Abstract

**Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank**
Semantic word spaces have been very useful but cannot express the meaning of longer phrases in a principled way. Further progress towards understanding compositionality in tasks such as sentiment detection requires richer supervised training and evaluation resources and more powerful models of composition. To remedy this, we introduce a Sentiment Treebank. It includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences and presents new challenges for sentiment compositionality. To address them, we introduce the Recursive Neural Tensor Network. When trained on the new treebank, this model outperforms all previous methods on several metrics. It pushes the state of the art in single sentence positive/negative classification from 80% up to 85.4%. The accuracy of predicting fine-grained sentiment labels for all phrases reaches 80.7%, an improvement of 9.7% over bag of features baselines. Lastly, it is the only model that can accurately capture the effect of contrastive conjunctions as well as negation and its scope at various tree levels for both positive and negative phrases.

**Paper:** Download pdf

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng and Christopher Potts

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

*Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*

**Dataset Downloads:**

Main zip file with readme (6mb)
Dataset raw counts (5mb)
Train,Dev,Test Splits in PTB Tree Format

**Code:** Download Page

**Press:** Stanford Press Release

Dataset visualization and web design by Jason Chuang. Live demo by Jean Wu, Richard Socher, Rukmani Ravisundaram and Tayyab Tariq.

This webpage requires one of the following web browsers:

Tweet | Like 26

README.md

# pysentimiento: A Python toolkit for Sentiment Analysis and Social NLP tasks

run_tests passing

A Transformer-based library for SocialNLP classification tasks.

Currently supports:

- Sentiment Analysis (Spanish, English)
- Emotion Analysis (Spanish, English)

Just do `pip install pysentimiento` and start using it:

Open in Colab

```
from pysentimiento import SentimentAnalyzer
analyzer = SentimentAnalyzer(lang="es")

analyzer.predict("Qué gran jugador es Messi")
# returns SentimentOutput(output=POS, probas={POS: 0.998, NEG: 0.002, NEU: 0.000})
analyzer.predict("Esto es pésimo")
# returns SentimentOutput(output=NEG, probas={NEG: 0.999, POS: 0.001, NEU: 0.000})
analyzer.predict("Qué es esto?")
# returns SentimentOutput(output=NEU, probas={NEU: 0.993, NEG: 0.005, POS: 0.002})

analyzer.predict("jejeje no te creo mucho")
# SentimentOutput(output=NEG, probas={NEG: 0.587, NEU: 0.408, POS: 0.005})
"""
Emotion Analysis in English
"""

emotion_analyzer = EmotionAnalyzer(lang="en")

emotion_analyzer.predict("yayyy")
# returns EmotionOutput(output=joy, probas={joy: 0.723, others: 0.198, surprise: 0.038, disgust: 0.01
emotion_analyzer.predict("fuck off")
# returns EmotionOutput(output=anger, probas={anger: 0.798, surprise: 0.055, fear: 0.040, disgust: 0.
```

Also, you might use pretrained models directly with `transformers` library.

# Vamos al Notebook 1

# ¿Cómo representar “matemáticamente” un texto?

# Tidy Text

No es la conciencia (...) la que determina su ser sino (...) el ser social lo que determina su conciencia.

| doc | word |
|---|---|
| 1 | no |
| 1 | es |
| 1 | la |
| 1 | conciencia |
| 1\| | la |
| 1 | que |
| 1 | determina |
| 1 | su |
| 1 | ser |
| ... | ... |

# Tidy Text

Un fantasma recorre Europa: el fantasma del comunismo

| doc | word |
| --- | --- |
| 2 | un |
| 2 | fantasma |
| 2 | recorre |
| 2 | europa |
| 2 | el |
| 2 | fantasma |
| 2 | del |
| 2 | comunismo |

# Tidy Text

No es la conciencia (...) la que determina su ser sino (...) el ser social lo que determina su conciencia.

Un fantasma recorre Europa: el fantasma del comunismo

| doc | word |
|---|---|
| 1 | no |
| 1 | es |
| 1 | la |
| ... | ... |
| 2 | el |
| 2 | fantasma |
| 2 | del |
| 2 | comunismo |

# Tidy Text

| doc | word |
|---|---|
| 1 | no |
| 1 | es |
| 1 | la |
| ... | ... |
| 2 | el |
| 2 | fantasma |
| 2 | del |
| 2 | comunismo |

group_by(doc) %>%
count(word)

| doc | word | count |
|---|---|---|
| 1 | no | 1 |
| 1 | es | 1 |
| 1 | la | 2 |
| 1 | conciencia | 2 |
| ... | ... | ... |
| 2 | el | 1 |
| 2 | fantasma | 2 |
| 2 | del | 1 |
| 2 | comunismo | 1 |

# Document-Term Matrix (TFM)

| doc | no | es | la | conciencia | ... | el | fantasma | del | comunismo |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 2 | ... | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | ... | 1 | 2 | 1 | 1 |

# Document-Term Matrix (TFM)

$$\text{Matriz } M = \begin{array}{cc} & \textit{F términos} \\ \rotatebox{90}{\textit{N documentos}} & \begin{bmatrix} V_{11} & V_{12} & V_{13} & \cdots & V_{1F} \\ & & & & \\ V_{N1} & V_{N2} & V_{N3} & \cdots & V_{NF} \end{bmatrix} \end{array}$$

**cada documento es una fila**

**cada columna está asociada a un término**



**Fuente:** https://github.com/gefero/ws_text_mining/blob/master/slides/04_pinto.pdf

# Document-Term Matrix (TFM)

Palabras, bigramas, trigramas, lemas, solo la raíz de la palabra...

$$\text{Matriz } M = \begin{array}{cc} & \textbf{\textit{F términos}} \\ \textbf{\textit{N documentos}} & \begin{bmatrix} V_{11} & V_{12} & V_{13} & \cdots & V_{1F} \\ & & & & \\ V_{N1} & V_{N2} & V_{N3} & \cdots & V_{NF} \end{bmatrix} \end{array}$$

Frecuencia del término

factor~data
IDAES_UNSAM

**Fuente:** https://github.com/gefero/ws_text_mining/blob/master/slides/04_pinto.pdf

# Bag of Words (BoW)

- Representación de cada documento en función de las palabras que contiene

- Características:
  - Es simple de generar
  - Se asume que las palabras son “independientes”
  - Los vectores son claramente no independientes
  - La gramática y el orden de las palabras se pierden

# Métricas (TF, IDF, TF-IDF)

- Podemos pensar en dos dimensiones de las frecuencias de los términos de un corpus…
    - Un término *t* es más **importante** si es más frecuente en un documento d de un corpus *C* determinado.
    - A su vez, *t* es más **informativo** del contenido de un documento *d* si está presente en pocos documentos y no en todos de *C* .

- Mirar tanto la frecuencia de *t* a lo largo de todo el corpus *C* y al interior del documento *d*.

# Métricas (TF, IDF, TF-IDF)

- $c(t,d)$es el conteo çrudo”del t en el documento d
- $rtf(t,d) = c(t,d)$
- Hasta aquí estamos en el esquema BoW crudo.

- Problemas:
  - El largo de los documentos suele ser variable
  - En general, la información acerca del sentido no çrece de forma proporcional a la ocurrencia de t en un d

- Entonces, hay normalizaciones alternativas
  - Binaria: 0, 1
  - $TF(t,d) = \frac{c(t,d)}{\sum c(t,d)}$
  - Log: $logtf = 1 + log(c(t,d))$

factor~data
IDAES_UNSAM

# Métricas (TF, IDF, TF-IDF)

- Insumo para una medida de la informatividad de un término a lo largo de C

$$DF(t) = log\frac{df(t)}{|C|}$$

Donde

- df (t) es la cantidad de documentos en C que contienen el término t
- |C| es el tamaño del corpus C , es decir, el total de documentos en C

- Cuanto mayor es DF (t) menor es la informatividad de un término. Entonces, se calcula su inversa (IDF):
  - $IDF(t) = log\frac{|C|}{df(t)}$

# Métricas (TF, IDF, TF-IDF)

- Entonces, tf (t, d) es una propiedad del documento y IDF (t) es una propiedad del corpus
- Combinamos ambas en una medida llamada Term Frequency-Inverse Document Frequency (TF-IDF)
    - $TFIDF(t) = tf(t,d) \times IDF(t)$

- Valores altos de tf (t, d) y valores altos de IDF (t) -o sea, valores bajos de DF (t) arrojan valores altos de TF IDF (t).
- O sea, términos t frecuentes en d y poco frecuentes en C .

# Vamos al Notebook 2