Introducción a Procesamiento de Lenguaje Natural:
Análisis de Sentimientos con KNN y PCA

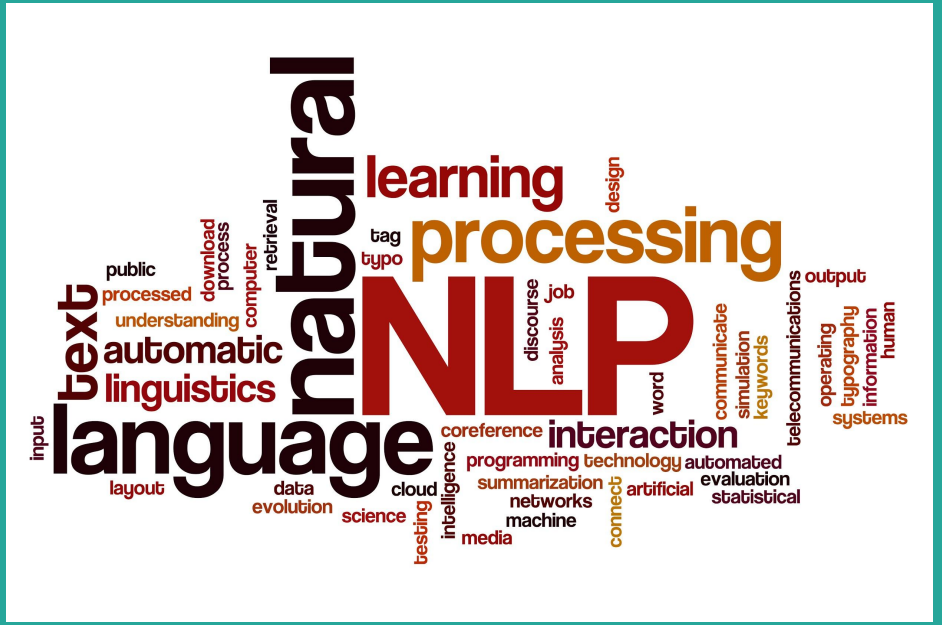
Métodos Numéricos

Procesamiento de Lenguaje Natural

Técnicas computacionales para procesar el lenguaje escrito para:

1. Entenderlo
2. Generarlo

Nombres relacionados:
Computational Linguistics, Text Mining



IMDB

- Sitio con base de datos de películas y reseñas de estas
- Paper de Maas et al (2011) recolectó un dataset de reseñas etiquetadas como negativas y positivas
- ¡Queremos armar un clasificador de reseñas positivas y negativas!

Análisis de Sentimientos

¿Es positivo o negativo el texto?

- Acabo de regalarle a mis sentidos una serie de momentos difíciles de olvidar; acabo de ver “El secreto de sus ojos” y es una película total. Casi que no me sale poner un comentario
- Si esta película ganó el Oscar todo es posible en este mundo. Este era mi otro título de la crítica, pero lo deseché porque pensaba que era muy largo para lo poco que decía.
- Juan José Campanella. Qué más se puede decir de este supuesto nuevo genio argentino. (...)

Proceso automático para entender el sentimiento u opinión de un texto.

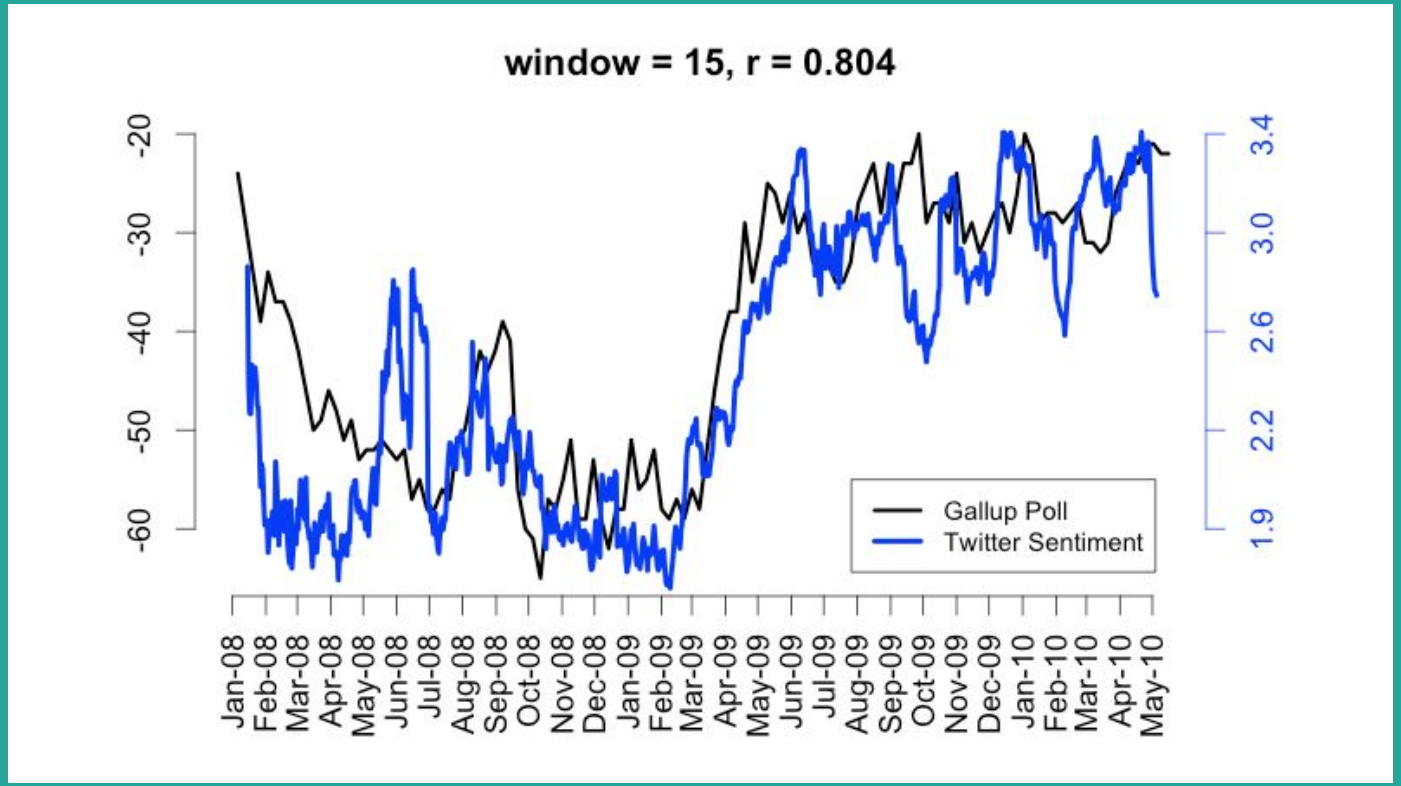
Tarea no trivial: no basta ver el léxico.

Solemos usar algoritmos estadísticos que “aprenden” a reconocer estos sentimientos.

Análisis de Sentimiento y Política

- ¿Qué candidato tiene más aceptación de sus votantes?
- ¿A qué político atacan más en las redes?
- ¿Cómo hacemos campaña personalizada para los votantes? (o sea, lo que hizo Cambridge Analytica)

Sentimiento Twitter vs Confianza Consumidores



(Robado de slides de curso de NLP de Dan Jurafsky)

Lo que vamos a hacer:

1. Convertir cada reseña en un vector de largo fijo
2. Usar un algoritmo de aprendizaje supervisado (kNN)
3. Entrenarlo con una parte de los datos
4. Corroborar la performance sobre otro conjunto (testing)
5. Utilizar PCA para ver si mejora performance
6. Experimentar con los parámetros de KNN y PCA

Paso previo: Preprocesado

- “Tokenizado”: separar el texto en “tokens”: unidad de texto que tenga sentido en nuestro problema
- Es un proceso no tan trivial
- También convertimos a minúscula y sacamos algunos signos de puntuación

Ejemplo:

“Sr. Simpson, su silencio lo incrimina más y más” ->

[“sr.”, “simpson”, “su”, “silencio”, “lo”, “incrimina”, “más”, “y”, “más”]

Modelo Bag of Words (BoW)

- Ignora el orden de las palabras
- Ordenamos nuestro vocabulario {w1, w2, …, w3}
- Cada texto se convierte en un vector de las ocurrencias de cada palabra

Ojo: si nuestro vocabulario es gigante estamos en problemas

Ejemplo:

V = (chicos, chicas, y, quieren, rock, falta, te)

“chicos y chicas quieren rock quieren rock”
-> (1, 1, 1, 2, 2, 0, 0)

“te falta rock”

-> (0, 0, 0, 0, 1, 1, 1)

Problema:

- El vocabulario generado tiene 160K palabras
- Palabras muy comunes no dan mucha información
- Las más infrecuentes no le permiten generalizar a nuestros algoritmos
- Compromiso: filtrar las más frecuentes y las más infrecuentes

Palabras más comunes:

The, and, a, of, to, is, br, it, in, was, with, movie, but, film

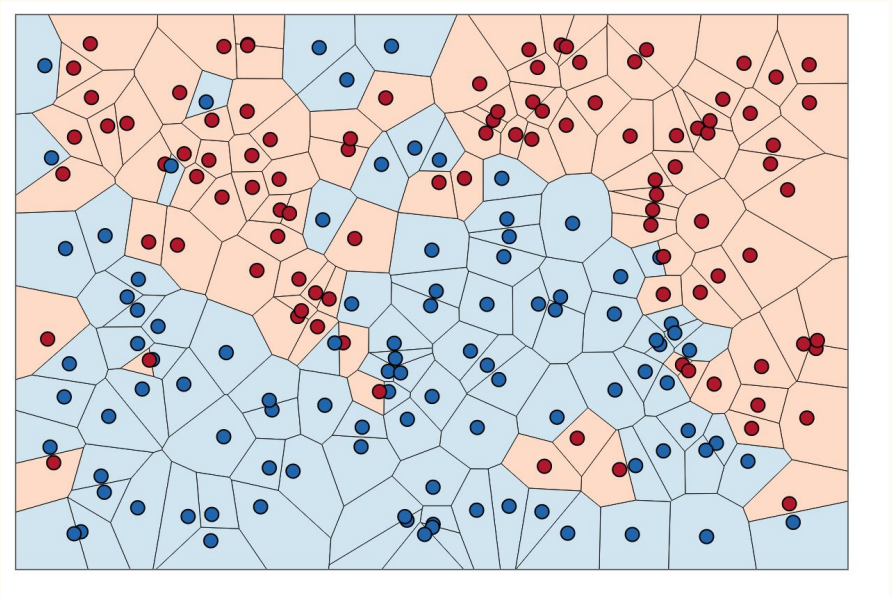
Palabras más infrecuentes:

insieme,giorni,quei,vanoni,grotesqueries,d'offizi,shock-lovers,argenta,pre-zombie,townswoman,moon/darkwave,alterns,near-delusional,bedouins,edifices,sandscapes,napoleanic

Clasificación

1. Ya tenemos vectores de largo **m**
2. Dos conjuntos de reseñas/vectores: Entrenamiento y Testing
3. Vamos a utilizar KNN: para cada reseña/vector de testing, busco los **k**-vecinos más cercanos.

   Le pongo la etiqueta mayoritaria entre estos k vecinos.
4. ¿Qué significa esto en nuestro dominio?



Implementar:

- Implementar clasificador de sentimientos usando kNN
- Implementar PCA

Experimentar:

- Experimentar con k = cantidad de vecinos. ¿Cómo afecta la performance de nuestro algoritmo?
- ¿En qué casos falla nuestro algoritmo?
- ¿PCA mejora la performance del algoritmo? ¿qué parámetro α es el mejor?
- Elegir mejor configuración en base a los resultados de los algoritmos en testing