IIC 3800 Tópicos en CC
NLP

https://github.com/marcelomendoza/IIC3800

- OUTLINE -

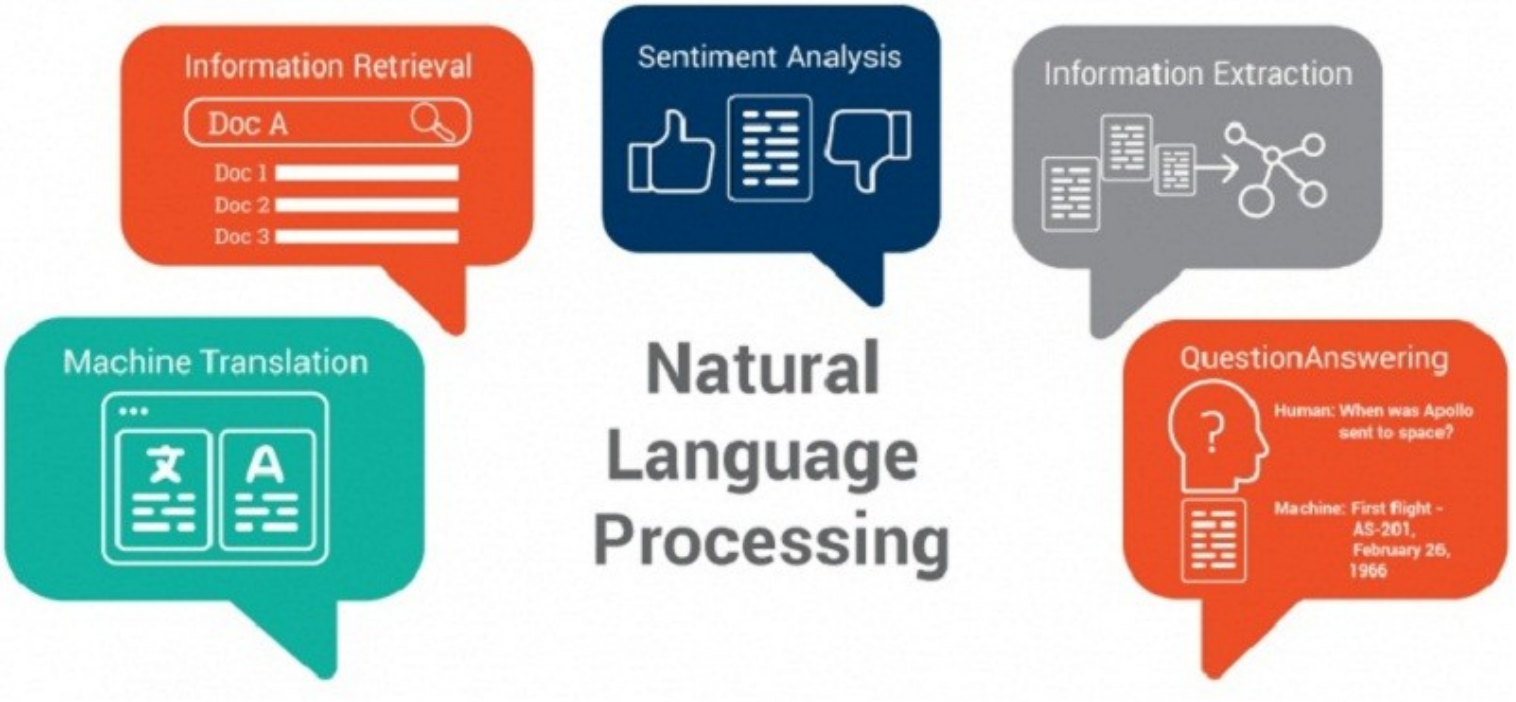
¿Qué vamos a ver?

- Ranking y clasificación de documentos
- Word2vec, Glove, ELMO, BERT
- Seq2seq, machine translation
- GPT, ChatGPT, chatbots
- NLP cross-lingual, multi-lingual, fairness

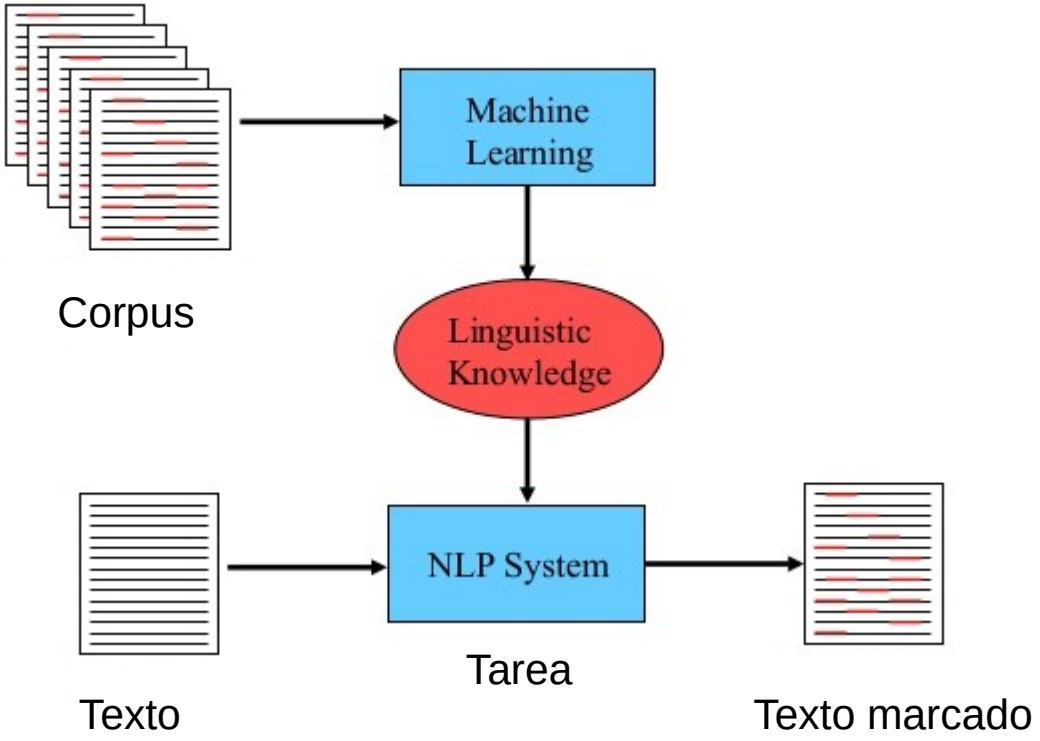
- INTRODUCCIÓN -

Introducción

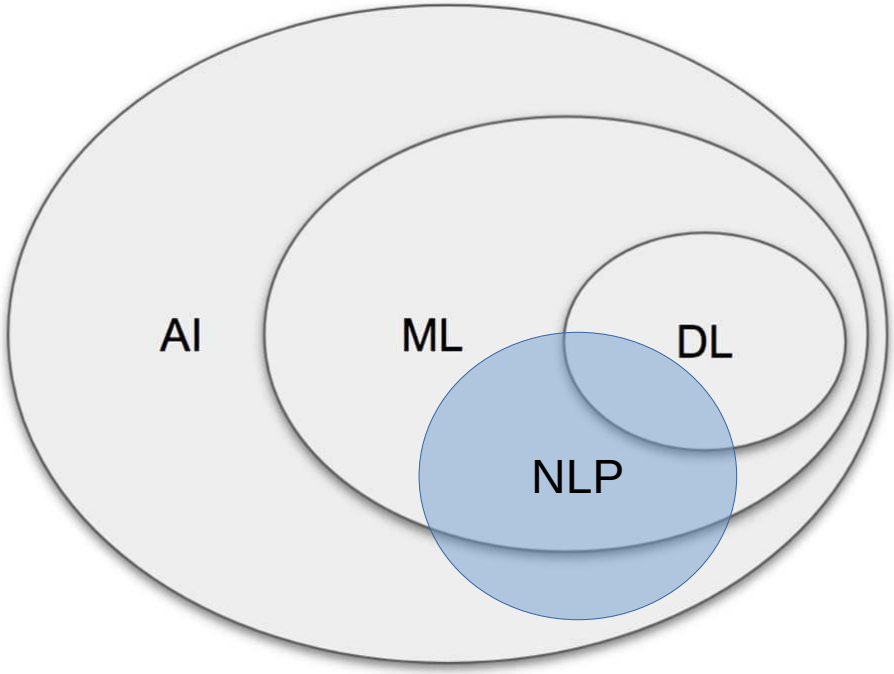
Síntesis. ¿Cuáles tareas aborda NLP?



El enfoque de NLP (clásico)

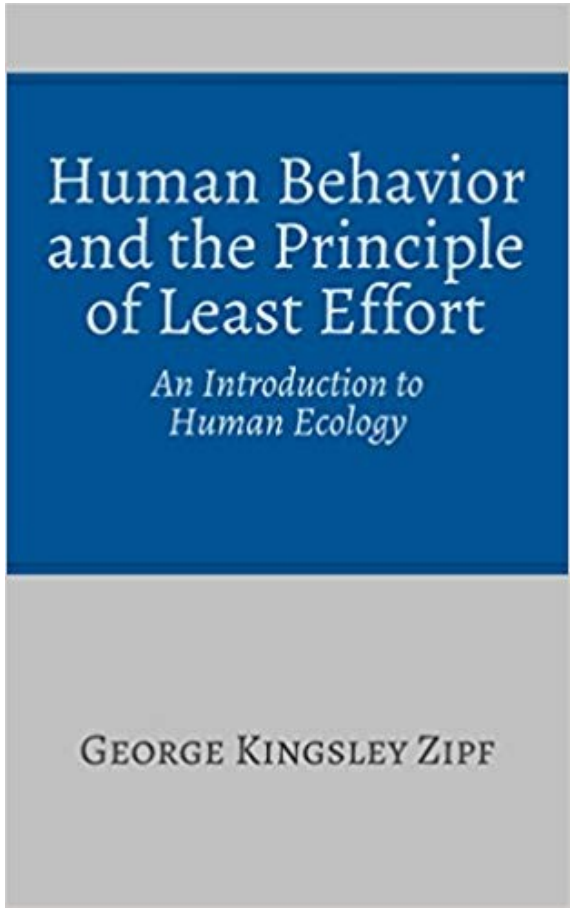


¿Dónde se ubica NLP?



- LEYES DEL TEXTO Y PROCESAMIENTO BÁSICO -

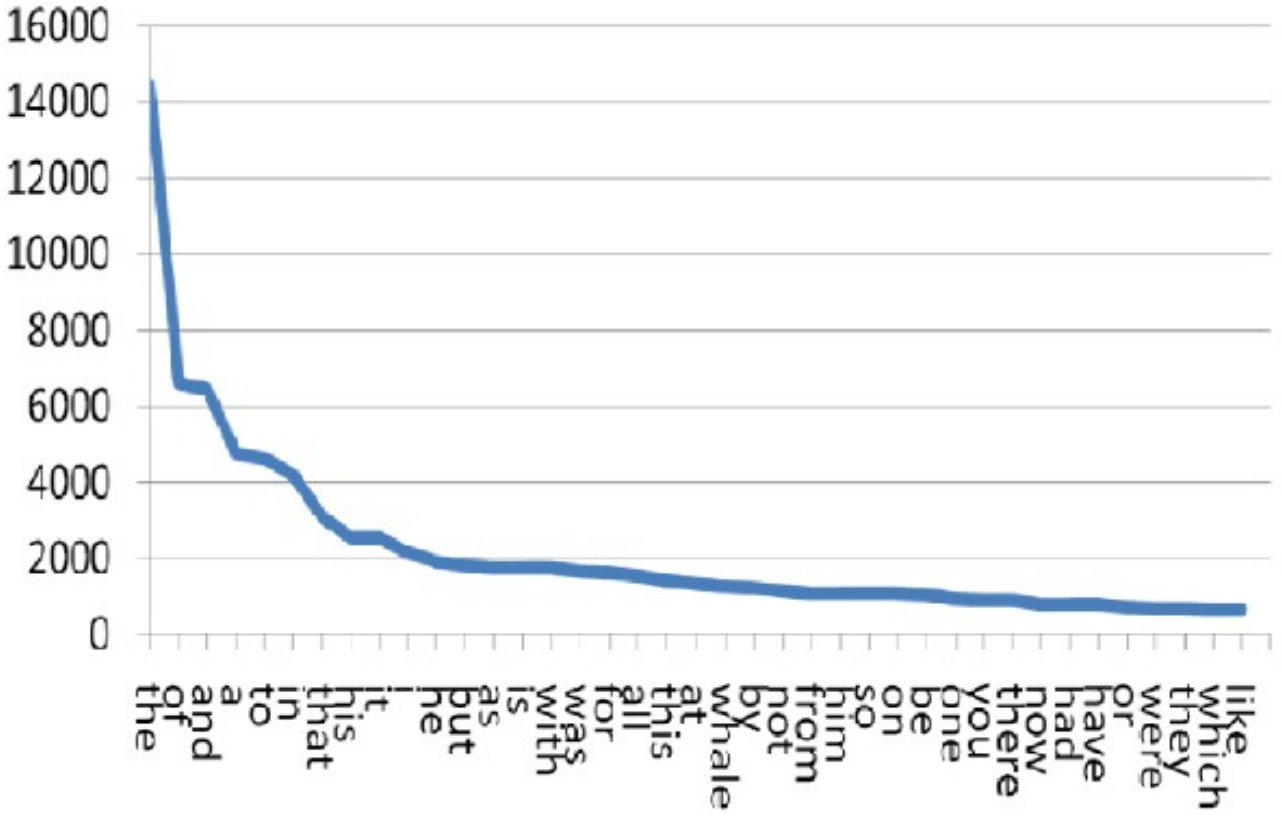
Leyes del texto



George Kingsley Zipf (1949), Human behavior and the principle of least effort, Addison-Wesley Press

Leyes del texto

Zipf para Reuters[2]



[2]Dataset de noticias, disponible on-line

Leyes del texto

Word	Freq. (f)	Rank (r)	$f \cdot r$	Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

Producto $f \cdot r$ en el libro *Tom Sawyer*, versión en inglés.

Leyes del texto

θ : pendiente de la curva log-log

n : # tokens

r : ranking de la palabra

f : # ocurrencias de la palabra

Ley de Zipf: $f \sim \frac{1}{r}$

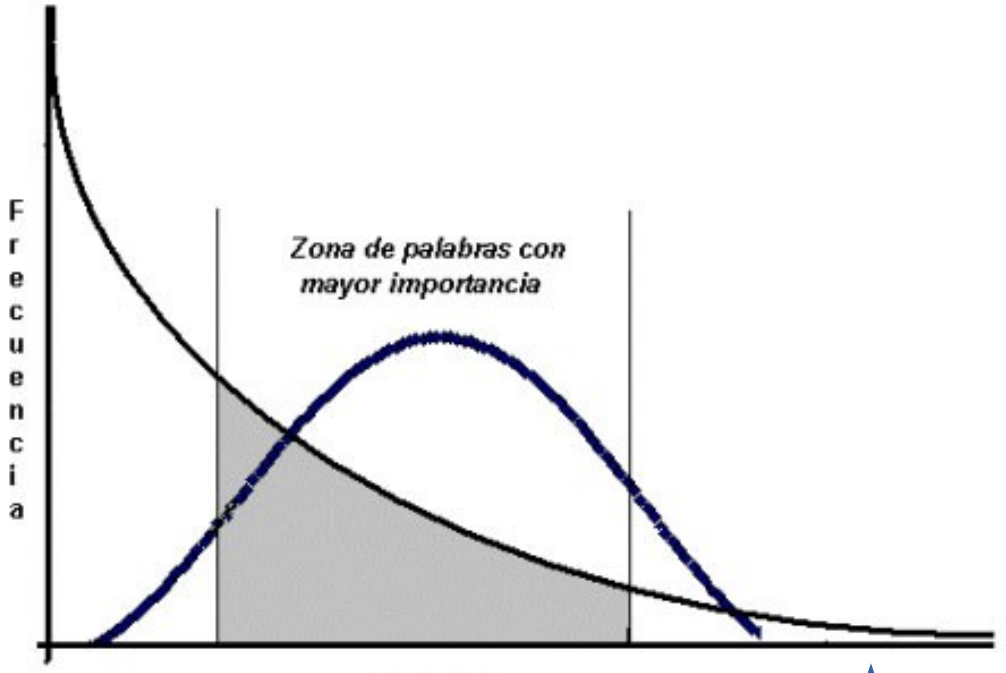
$$f \sim \frac{1}{r^\theta}$$

$$f_r = \frac{n}{r^\theta \cdot H_V(\theta)}$$

Si $\theta \approx 1 \rightarrow H_V(\theta) = \log(n)$

$$H_V(\theta) = \sum_{j=1}^{V} \frac{1}{j^\theta}$$

Leyes del texto

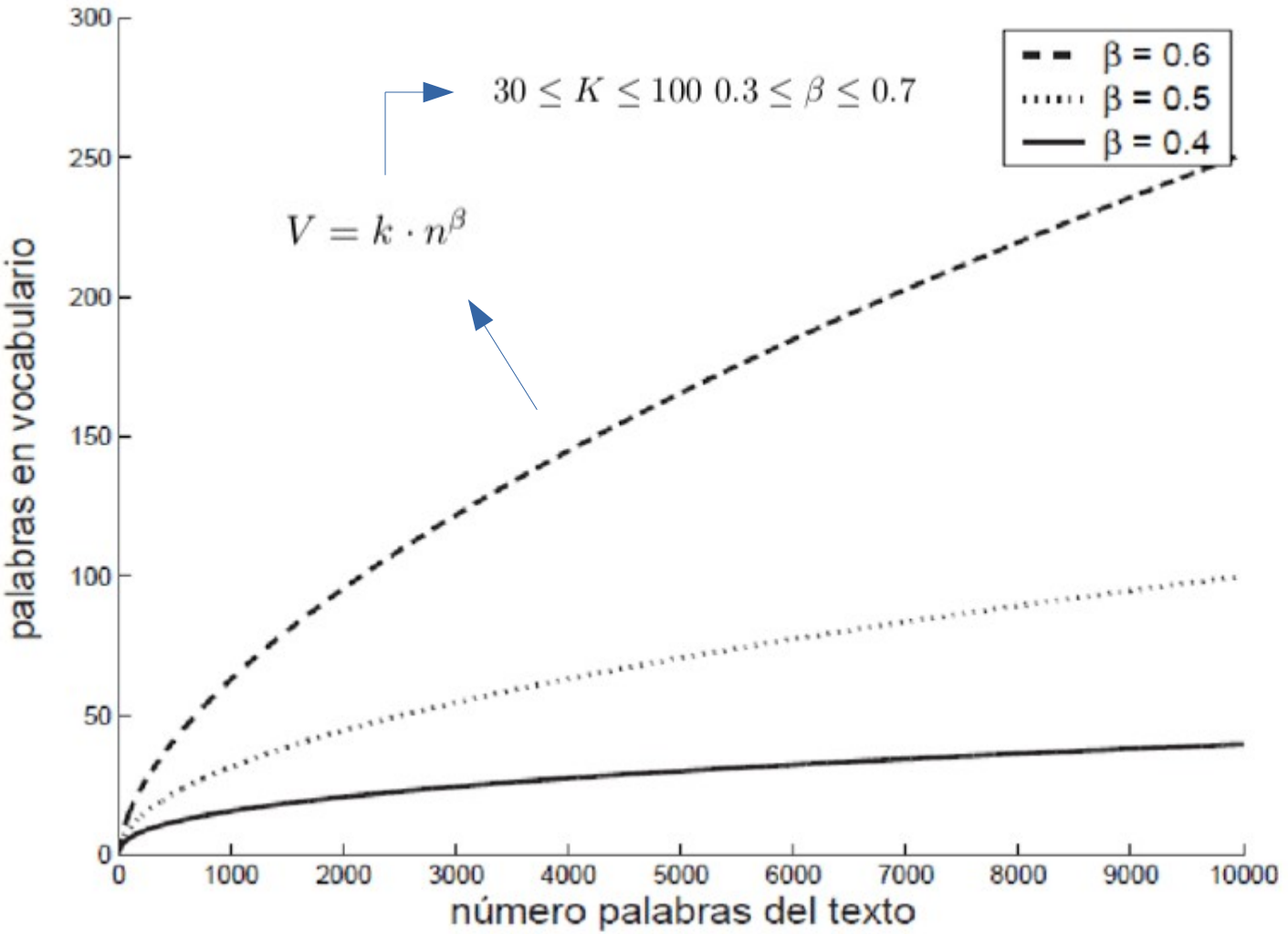


stopwords

keywords

outliers (e.g. typos)

Leyes del texto (Heaps)



Procesamiento básico

- Token – String delimitado que aparece en el texto.
- Término – token con significado según un corpus (por ejemplo diccionario)
- Input:
  amigos, Romans, habitantes. | habia una vez ... Cesar | ...
- Output:
  amigo | romano | habitante | cesar | ...
- Cada token es candidato a término.
- Cuáles elegimos? Depende del corpus.

Procesamiento básico

Lematización

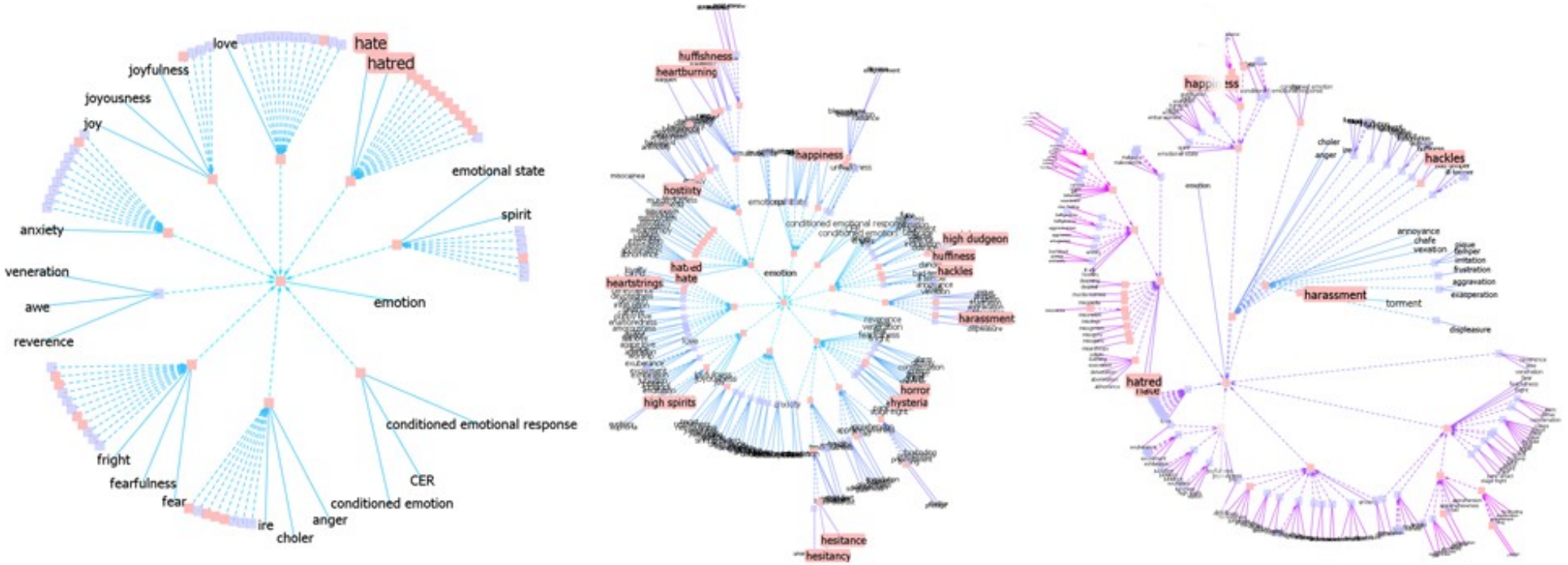
WordNet lemmatizer

- Reducir formas inflexionales a su raíz → Raíz semántica
- Ejemplo: *am, are, is* → *be*
- Ejemplo: *autos, auto, automoviles* → *auto*
- Ejemplo: *Los autos de los jóvenes son de colores* → *auto joven es color*
- Lematización implica realizar una reducción hacia la raíz (lema). (*destruccion* → *destruir*)

Procesamiento básico

WordNet es una enorme red de palabras

- 155287 palabras organizadas en 117659 synsets



Zoom out → Zoom in →

- Ejemplos -

Preprocesamiento de texto en NLTK

Procesamiento básico Web:

```
> import nltk
> from urllib import urlopen
> url = "http://www.gutenberg.org/files/2554/2554.txt"
> raw = urlopen(url).read()
```

Tokenización y creación del objeto texto:

```
> tokens = nltk.word_tokenize(raw)
> text = nltk.Text(tokens)
```

Ahora podemos hacer NLP sobre el texto:

```
> text.collocations()
> ...
```

Preprocesamiento de texto en NLTK

Procesamiento de HTML:

```
> url = "http://nltk.org"
> html = urlopen(url).read()
> raw = nltk.clean_html(html)
```

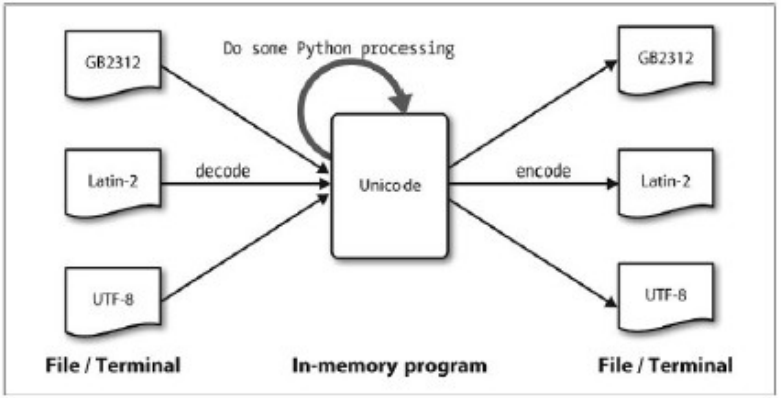
Repetimos el pipe anterior:

```
> tokens = nltk.word_tokenize(raw)
> text = nltk.Text(tokens)
> text.collocations()
```

Construir el vocabulario (minúsculas y sorted set):

```
> words = [w.lower() for w in text]
> vocab = sorted(set(words))
```

Leer con decode, procesar en Unicode, print con encode (render glyphs).



Procesamiento de Unicode (Spanish!):

```
> url = "http://www.inf.utfsm.cl"
> html = urlopen(url).read()
> raw = nltk.clean_html(hmtl)
> decoded = raw.decode('utf8')
> print decoded.encode('latin2')
```

Preprocesamiento de texto en NLTK

Stemmers:

```
> porter = nltk.PorterStemmer()
> lancaster = nltk.LancasterStemmer()
> [porter.stem(t) for t in tokens]
> [lancaster.stem(t) for t in tokens]
```

Lematizador (stemmer + corpus checking):

```
> wnl = nltk.WordNetLemmatizer()
> [wnl.lemmatize(t) for t in tokens]
```

Preprocesamiento de texto en NLTK

Segmentador para texto raw en inglés:

```
> url = "http://www.gutenberg.org/files/2554/2554.txt"
> raw = urlopen(url).read()
> sent_tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
> sents = sent_tokenizer.tokenize(raw)
```

Entrega una lista de sentencias:

```
> len(sents)
> print sents[1].encode(latin2)
```

Podemos mejorar el tokenizer de NLTK, agregando expresiones regulares que queremos detectar como unigramas:

```
> pattern = r'''(?x)
...     ([A-Z]\.)+          # abreviaciones (U.S.A.)
...   | \w+(-\w+)*          # palabras con guiones
...   | \$?\d+(\.\d+)       # precios
...   | \.\.\.              # elipsis
...   | [][.,;"'?():-_]     # tokenizadores
... '''
>>> nltk.regexp_tokenize(text,pattern)
```

https://github.com/marcelomendoza/IIC3800